VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vizualizace mozkové aktivity

Brain Activity Visualization

2014

Bc. Pavel Ferdian

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Pavel Ferdian**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Vizualizace mozkové aktivity
Brain Activity Visualization

Zásady pro vypracování:

Cílem práce je implementovat aplikaci pro vizualizaci mozkové aktivity. Bude využito zařízení EMOTIV EPOC Headset. Zvolenou technologií je Microsoft .NET a jazyk C#.

1. Seznámení se s problematikou snímání a vizualizace EEG signálů. Přehled existujících technologií s ohledem na další použitelnost zařízení a software v nemedicínských aplikacích.
2. Návrh a implementace aplikace s požadovanou funkcionalitou.
3. Výkonnostní testy a experimenty s reálnými daty, uživateli.
4. Vyhodnocení experimentů a dosažených výsledků.

Seznam doporučené odborné literatury:

[1] John Kessenich: The OpenGL Shading Language, http://www.opengl.org/documentation/glsl/, 2011
[2] Mark Segal and Kurt Akeley: The OpenGL Graphics System: A Specification, 2011
[3] Saeid Sanei, J. A. Chambers: EEG Signal Processing, Wiley-Interscience; 1 edition (September 11, 2007), ISBN-13: 978-0470025819

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Gajdoš, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 30. dubna 2014 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2014 .....

Tímto bych chtěl poděkovat vedoucímu diplomové práce Ing. Petru Gajdošovi, Ph.D. za cenné rady, připomínky a metodické vedení této práce. Dále bych chtěl poděkovat všem zúčastněným, za pomoc a ochotu v průběhu vzniku práce a při testování.

Poděkování

Tato práce byla vypracována s podporou projektu Bio-inspirované metody: věda, vzdělávání a transfer znalostí, reg. č. CZ.1.07/2.3.00/20.0073 podpořeného Operačním programem Vzdělávání pro konkurenceschopnost, financovaného ze strukturálních fondů EU a státního rozpočtu ČR.

Abstrakt

Cílem této práce bylo vytvořit aplikaci pro správu headsetu EPOC společnosti Emotiv. Úkolem této práce bylo také získat patřičné poznatky, které by umožnily pochopení funkcionality BCI systémů, jakožto zařízeních schopných umožnit ovládání elektronických zařízení pomocí lidské mysli. Výstupem práce jsou tři aplikace, sloužící jako obsluha headsetu EPOC, a zároveň prezentující reálné využité tohoto BCI systému. Posledním bodem práce, byla sada pokusů, vyhodnocující schopnosti BCI sytému EPOC a k němu dodávaného SDK.

Klíčová slova: Mozek, BCI, Emotiv, EPOC, EEG, WPF

Abstract

The goals of the thesis are creating of apllication for managment of EPOC headset from Emotiv company and making knowledge of funcionality BCI systems. Due to theses systems we are able to controle electronical machine by our mind. Results of this thesis are 3 aplliacations, which are presenting real use of these systems. Last point of the thesis was a colleciton of experiments. In these experminets make summary of abilities of BCI system EPOC and delivered SDK.

Keywords: Brain, BCI, Emotiv, EPOC, EEG, WPF

Seznam použitých zkratek a symbolů

BCI	–	Brain-Computer interface
XAML	–	Extensible Application Markup Language
WPF	–	Windows Presentation Foundation
SDK	–	Software development kit

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

S příchodem moderní vědy se objevily stroje a zařízení, které bylo nutné nějak ovládat. Nejdříve se ovládaly manuálně a s příchodem počítačů automaticky. Později se objevovaly metody ovládání za pomoci hlasových povelů, nicméně se v poslední době začíná rozmáhat teorie, která říká, že by se v blízké době mohly stroje ovládat pouhou myšlenkou. Ovládat cokoli pouhou představou může znít jako námět z nějakého laciného sci-fi románu, ale s příchodem moderní encefalografie se tato teorie pomalu mění v praxi. První zmínky o tom, že lidský mozek produkuje elektromagnetické vlnění, se objevily už na přelomu 19. a 20. století. Později se prokázalo, že toto vlnění lze asociovat k nějakému podnětu, a to vedlo k oné myšlence, že by šly stroje ovládat myslí. Vzniklo nové odvětví vědy, které se zabývá systémy založené na EEG, které by sloužily jako rozhraní mezi lidským mozkem, tedy myslí, a počítačem. Tomuto rozhraní se začalo říkat BCI (Brain-Computer interface). V posledních pár letech se začaly objevovat BCI systémy, které cílí na oblast široké veřejnosti, a tudíž už nejsou BCI systémy pouze záležitostí výzkumných institutů a společností.

Obsahem této práce bylo zjistit, jak si takové dostupné BCI zařízení vede. Testovaným produktem bylo cenově dostupné zařízení EPOC společnosti Emotiv, které nabízí přenosné řešení systému EEG. Cílem práce bylo pochopit, jak takovéto zařízení funguje, a co vše se s ním dá dělat. To vyžadovalo jisté znalosti a pochopení funkce lidského mozku. Dalším úkolem bylo vytvořit sadu aplikací, které by umožnily zařízení EPOC spravovat, a které by prezentovaly jeho reálné využití. Dalším nárokem kladeným na tuto práci bylo v rámci aplikace vizualizovat stav aktivity mozku, a to jako vizualizaci na modelu mozku. Posledním úkolem bylo vytvořit sadu pokusů, které měly ukázat schopnosti zařízení EPOC v rámci detekce kognitivních akcí.

2 Teoretické a technické zázemí práce

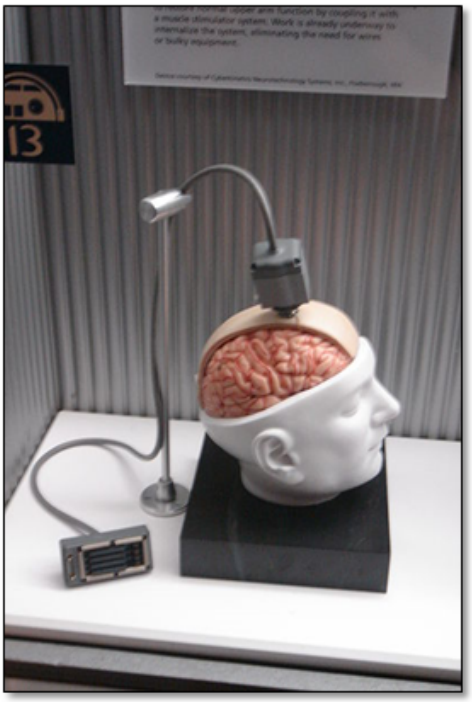
Náplní této práce bylo spravovat náhlavní soupravu EPOC, která se řadí do skupiny zařízení umožňující takzvaný BCI systém. Následující kapitola bude popisovat základní principy BCI systémů a samotné zařízení EPOC.

2.1 BCI

BCI (Brain-Computer interface) je rozhraní, sloužící k propojení elektronického zařízení snímajícího aktivitu mozkové činnosti s počítačem. Na BCI lze pohlížet jako na nástroj, ve kterém akce jedince neprocházejí obvyklými výstupy z mozku, jako jsou pohyb ruky, ale lze tak díky získaným datům z mozku provádět vnější podněty či aktivity. Primárně se BCI systémy využívají ve snaze pomoci lidem s tělesným postižením. Příkladem BCI systému je EEG (Elektroencefalogram). Existují tři typy BCI systémů. [2]

2.1.1 Invazivní metody BCI

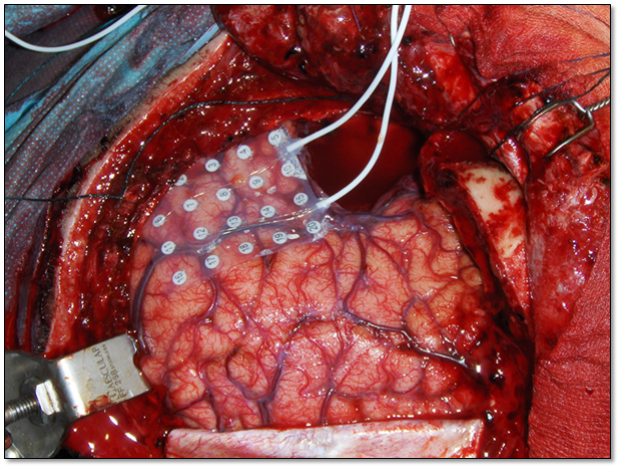
Prvním typem jsou plně invazivní systémy, kde je elektronika snímající aktivitu mozku implantována přímo do šedé kůry mozkové. Tento postup je velice riskantní, protože vede k neurochirurgickému zákroku a následně hrozí náchylnost k rozšiřování zjizvené tkáně mozku po zákroku. Tyto BCI systémy umožňují získávat velice kvalitní signál. [2, 4]



Obrázek 1: Vyobrazení modelu invazivního systému BCI

2.1.2 Částečně invazivní systémy BCI

Druhým typem jsou částečně invazivní BCI systémy, kde se na rozdíl od plně invazivních systémů neimplantují do šedé kůry mozkové, ale jen na vnitřní povrch lebky a zbytek systému je vyveden mimo lebku ven. Zákrok implantující částečně invazivní BCI systém je méně riskantní, ale kvalita získaného signálu mozkové aktivity je o něco menší než u plně invazivních systémů, nicméně je stále kvalitnější než u systémů neinvazivních. [2, 4]



Obrázek 2: Aplikace částečně invazivního systému BCI

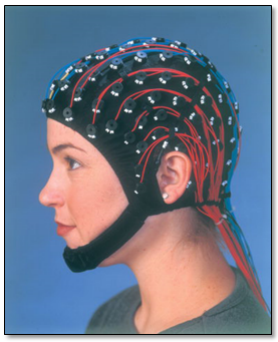
2.1.3 Neinvazivní systémy BCI

Třetím a běžně používaným typem jsou systémy neinvazivní. U těchto systémů nedochází k žádným výrazným lékařským zákrokům. Snímání mozkové aktivity se provádí například pomocí technologie EEG. Snímače (elektrody) jsou umístěny přímo na hlavě. Kvalita signálu je oproti předchozím metodám znatelně horší, což je zapříčiněno deformací průchodu napěťového potenciálu neuronu skrz tkáň lebky a kůže. [2, 4]

Příkladem využití BCI systémů je usnadnění ovládání různých zařízení lidem s tělesným postižením nebo možnost spojení se systémem rozšířené reality, což znamená zcela nový přístup ovládání.

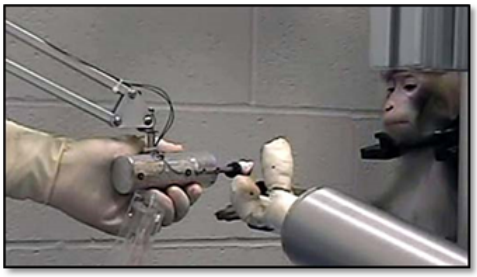
2.1.4 Pokusy na zvířatech a lidech

Příchod systémů BCI přinesl řadu zajímavých experimentů a pokusů. Jedním z nejzajímavějších pokusů byl pokus zavedení BCI systému do mozku opice. V tomto experimentu dokázal subjekt ovládat robotickou ruku, a tak se za pomoci kortikální motorické aktivity dokázal sám krmit. Robotická ruka se pohybovala v tří-dimenzionálním prostoru, a její zakončení, jež tvořil uchopovací mechanizmus, umožňovalo zachytit a udržet subjektem



Obrázek 3: Ukázka neinvazivního systému BCI

vybraný objekt, v tomto případě stravu. Experiment byl veden v Schwartz lab, University of Pittsburgh, USA. Video z experimentu. Zdroj youtube.com [2]



Obrázek 4: Experiment: opice ovládající robotickou ruku (Zdroj: [2])

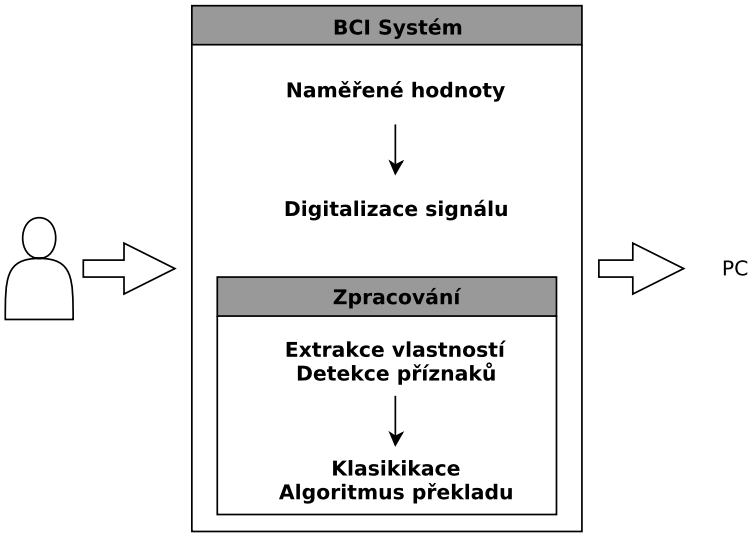
Jedním z průlomových experimentů bylo bezesporu propojení BCI systému s člověkem. Subjektem byl tělesně hendikepovaný Tim Hemmes, který mohl pomocí myšlenek nejprve hýbat kursorem na monitoru počítače, a následně i reálnou robotickou rukou. Projekt s názvem Brain-Computer Interface tak znamenal obrovský úspěch v implementaci BCI systému. Video z experimentu. Zdroj youtube.com [2]

2.1.5 Princi fungování BCI systémů

Každý BCI systém má vstup reprezentovaný vodiči zakončenými elektrodami napojenými na mozek, a to buď invazivně, nebo neinvazivně. Dále má výstup, který je pak pro běžné zařízení jako počítač dostatečně srozumitelný. BCI se tedy stará o to, aby byly získané analogové signály zpracovány dostatečně rychle, efektivně a pokud možno správně. BCI systém lze obecně, mimo vstup a výstup, rozčlenit do několika dílčích sekcí. Řekněme, že na vstupu máme surový EEG signál, který se nejprve podrobí vzorkování a následně filtraci. Na výstupu je pak diskrétní signál, se kterým se dále pracuje. Filtrace se provádí z důvodu přítomnosti nechtěného rušení, nebo z důvodu zájmu jen o některé části signálu.

Obrázek 5: Experiment: Time Hemes podává robotickou ruku ženě (Zdroj: [2])

Získaný signál se dále podrobí analýze a klasifikaci, jež jsou postupy vedoucí k rozpoznání jednotlivých objektů. Za objekt se může obecně považovat nějaká kognitivní či jiná hledaná činnost, ke které se přiřadí nějaká konkrétní aktivita. Klasifikace má dvě části, a to vznik příznaků a jejich následná klasifikace. Příznak si lze představit jako podmínku nebo vlastnost, která pomáhá objekty od sebe rozlišit. Příznak může vznikat mnoha způsoby, obecně ale existují dva běžně používané postupy. Prvním postupem vytváření příznaků je analýza problému, a následná definice rozhodovacího pravidla. Druhý postup, zvaný učení, vede k sestavení pravidla za pomocí již klasifikovaných objektů. Objekt je popsán příznakovým vektorem, což je soubor příznaků, popisujících jeho vlastnosti. Čím více příznaků vektor obsahuje, tím je objekt lépe popsán a přesněji rozpoznám, ale s větším množstvím příznaků roste také složitost, jak časová tak paměťová.



Obrázek 6: Princip BCI systému

Každý objekt je reprezentován třídou, do které se pomocí příznaků snaží získaný signál přiřadit, a tím určit vazbu mezi signálem a objektem. Tuto činnost provádí klasifikátor, což je stroj, procházející získané příznaky objektu. Ty přiřadí do tříd a analyzuje jejich pozici vůči příznakovému vektoru klasifikovaného objektu (třídy). Tato metoda se nazývá Minimální vzdálenost od těžiště třídy. Existují i další metody klasifikace, jako například metoda nejbližších sousedů, která je zobecněním metody Minimální vzdálenosti.

Existují i další postupy analyzující a vyhodnocující signály, kde většina z nich využívá pokročilých metod, jako jsou neuronové sítě. Neuronová síť je výpočetní model napodobující chování neuronů v mozku. Dokáže se učit a následně vyhodnocovat výsledky v závislosti na vstupních datech. Neuronové sítě se vyloženě hodí pro použití v BCI systémech. Učení neuronové sítě si lze představit jako vytváření a analýzu příznaků, a její celek jako klasifikátor. Neuronová síť má tu výhodu, že pokud se správně naučí, její vyhodnocování vede obvykle ke správným závěrům.

Výstupem BCI systému je nějaká konkrétní akce, kterou lze pak snadno převést na jakýkoliv úkon.

2.2 Emotiv EPOC

Emotiv EPOC je zařízení společnosti Emotiv, která se zabývá vývojem neinvazivních BCI systémů, založených na systému EEG. EPOC je bezdrátová náhlavní souprava (headset) pro snímání signálu EEG a je vybavena 14ti senzory a dvěma referenčními kanály. V balení se nachází radiový přijímač a set výměnných kontaktů. Samotný headset je vybaven lithiovou baterií, které má podle specifikace udržet zařízení v chodu až na 12 hodin. Headset také obsahuje gyroskop. Přenos dat mezi headsetem a přijímačem je přenášen bezdrátově šifrovaným signálem, a jako přijímač slouží USB dongle.

Emotiv EPOC je dodáván v krabici obsahující samotný bezdrátový headset, USB přijímač signálu, jednu sadu šestnácti výměnných kontaktů a nádobu s hydratačním roztokem. Společnost Emotiv umožňuje dokupovat náhradní příslušenství a dodatečný software ve svém internetovém obchodě. Omezením je dostupnost náhradních dílů, které lze získat jen v tomto internetovém obchodě.

Obrázek 7: Emotiv EPOC (Zdroj: www.emotiv.com)

2.2.1 Vlastnosti

Headset EPOC je vybaven 14-ti senzory rozmístěnými na plastových ramenech tak, aby každý senzor byl namapován na užitečnou část mozku. Tyto senzory jsou rozděleny do dvou skupin. Aktivní senzory AF3, AF4, F3, F4, F7, F8, FC5, FC6, P7, T7, O1, O2 a referenční P3 a P4.

Rozdělení senzorů podle oblastí mozku

- Čelní lalok - AF3, AF4, F3, F4, F7, F8
- Temenní lalok - P7, P8
- Týlní lalok - O1, O2
- Temporální lalok - FC5, FC6, T7, T8

Samotný signál je filtrován, aby byl zbaven nechtěných artefaktů či rušení. Vzorkovací frekvence headsetu je rovna 2048 Hz. Tento signál je následně převzorkován na 128Hz a je podroben hardwarovému filtrování třemi filtry.

Hardwarové filtry:

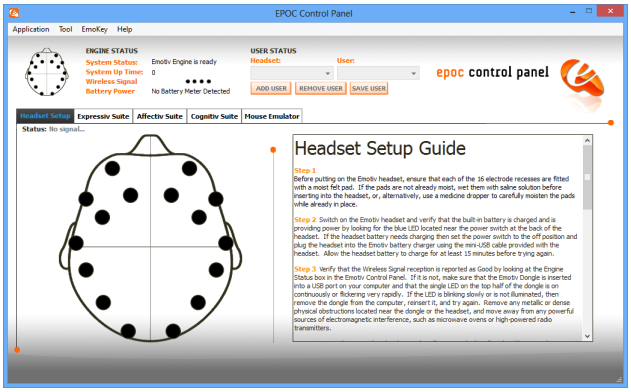
- High-pass filtr na 85Hz
- Low-pass filtr na 0.16Hz
- Notch filtr na 50 Hz a 60 Hz

2.2.2 Emotiv EPOC SDK

Emotiv EPOC SDK obsahuje sadu aplikací prezentujících základní schopnosti headsetu. Příkladem jsou tři aplikace znázorňující konkrétní interakce.

- Expressiv Suite – zpracování expresivních výrazů tváře
- Affectiv Suite – zpracování afektivních stavů (emocí)
- Cognitive Suite – zpracování myšlenek
- Hra Spirit Mountain - prezentující využití headsetu
- EmoKey – aplikace umožňuje namapovat klávesy klávesnice na kognitivní akce
- EmoComposer - aplikace sloužící k simulaci headsetu

SDK také obsahuje knihovny pro vlastní implementaci aplikací využívajících headsetu EPOC. Knihovny jsou primárně určeny pro vývoj v jazyce c++, ale součástí SDK jsou i wrappery pro jazyky java a c#. Existuje i několik neoficiálních implementací wrapperu, rozšířené o některé další funkce. OpenSource verze knihoven SDK se nazývá emokit.



Obrázek 8: EPOC Control Panel

Vývojářské SDK je tvořeno knihovnami edk.dll a edk_utils.dll, obsahující všechny potřebné prostředky pro obsluhu a získání dat z headsetu. Knihovna edk.dll obsahuje základní třídu EmoEngine, která má za úkol shraňovat data a informace z headsetu, a ty zpracovávat do srozumitelné formy. Jsou zde integrovány základní učící a vyhodnocovací algoritmy.

Společnost Emotiv nenabízí své SDK jako aktualizace, a to znamená, že každou novou verzi je nutné zakoupit znovu. SDK je dostupné pro operační systémy Microsoft Windows, Linux a Apple Mac OS X.

2.2.3 Výhody a nevýhody

Každé zařízení má své výhody a nevýhody, a není tomu jinak ani u headsetu Emotiv Epoc. Mezi výhody lze zařadit cenu tohoto zařízení, která je velice zajímavá, vzhledem k tomu co toto zařízení umí. Jak se později ukázalo největší nevýhodou zařízení EPOC jsou výměnné kontakty. Díky tomu, že se tyto kontakty musí před použitím namáčet do fyziologického roztoku, což je vlastně několika málo procentní roztok Chloridu sodného (NaCl), dochází k usazování sražené soli na pozlacený plíšek kontaktu a jeho následné oxidaci. To vede k nenávratnému poškození jemných plastových částí kontaktů, jako jsou bajonetové závity. Další nevýhodou je právě nutnost kupovat náhradní díly v oficiálním internetovém obchodě, navíc mnohdy s omezením, kdy je nutné pro státy mimo USA objednávat větší množství položek.

2.2.4 Alternativy k headsetu Emotiv EPOC

Emotiv EPOC není jedinou náhlavní sestavou umožňující BCI. Příkladem je náhlavní sestava MindWave a MindSet od společnosti Neurosky. Jak MindWave tak MindSet opět využívají systému EEG a na rozdíl od EPOCu mají pouze jeden jediný senzor namapovaný na čelní lalok mozku, který ale není potřeba namáčet do fyziologického roztoku. MindSet je na rozdíl od MindWave koncipován jako sluchátka, a to z důvodu odrušení vnějších vlivů a naopak zdůraznění sluchových vjemů.

Obrázek 9: Headsety MindSet a MindWave (Zdroj: www.neurosky.com)

2.3 Princip funkce neinvazivního Systému BCI EEG

Neinvazivní systémy využívají přístroje EEG a jejich princip je vyobrazen na obrázku 3. První zmínky o elektromagnetické aktivitě mozku se objevily v roce 1924, kde H. Berger poprvé registroval bioelektrickou aktivitu mozku. První EEG přístroje se pak objevily ve 30. letech 20. století a od té doby se stále vyvíjejí a hojně využívají.

EEG v principu detekuje a zpracovává bioelektrickou aktivitu mozku, kde elektrická aktivita je přesun elektricky nabitých iontů při změnách vodivosti buněčných membrán. Tyto změny produkují elektromagnetický potenciál, který senzory EEG zachycují. Zachycená jednotka, zvaná dipól, je tvořena dvěma stejně velkými opačně nabitými náboji.

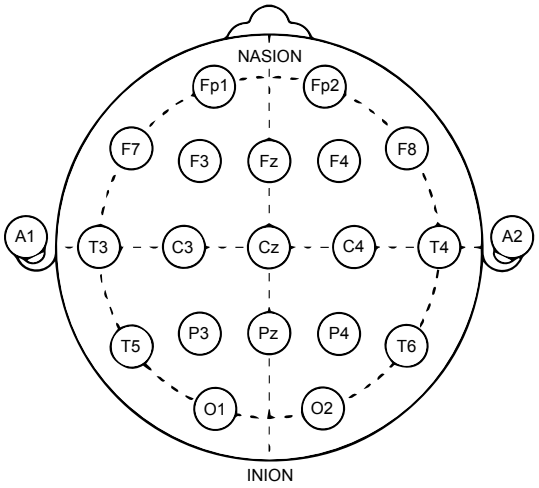
Samotný EEG přístroj je tvořen čtyřmi základními částmi. Elektrodami, předzesilovačem, zesilovačem a filtrem. Elektrody jsou kontakty dotýkající se konkrétních bodů lidské hlavy a z důvodu lepšího a kvalitnějšího přenosu signálu, je mezi kontakt a pokožku hlavy nanášeno medium, což je například fyziologický roztok nebo gel. Elektrody získávají elektromagnetický potenciál v daném bodě, který následně přenášejí do zesilovací části EEG.

Z důvodu velmi malé velikosti získaného potenciálu je nutné tento potenciál dostatečně kvalitně zesílit. K tomuto účelu slouží předzesilovač, který zesílí signál z řádů mikrovoltů na milivolty. Signál se následně ještě zesiluje zesilovačem, jehož výstupem je signál rovný rozdílu potenciálu na dvou elektrodách vstupující do zesilovače.

Další částí EEG zařízení je filtrace. Ta se provádí pomocí filtru dolní a horní propustě. Signál se filtruje z důvodu přítomnosti nežádoucích artefaktů a signálů jako jsou například svalová aktivita, ale i artefakty a rušení vznikající při přenosu signálu. Tyto filtry musí být dostatečně rychlé a spolehlivé. Výsledný analogový signál se buď přenáší do výstupního zařízení, například do tiskárny, nebo se převádí do digitální formy [1].

Moderní neinvazivní systémy využívají signály EEG, které se převádějí do srozumitelnější formy. S ohledem na proměnlivost charakteristik vlastností měnících se v závislosti na čase, a na různorodost chování mozku u každého člověka, není převod snadný. Nelze tedy s naprostou jistotou říci, že algoritmus převodu fungující u člověka A bude fungovat i u člověka B. Bylo tedy potřeba vyvinout postupy, které částečně kompenzují měnící se vlastnosti mozku. V těchto algoritmech se využívají takzvané Neuronové sítě.

V minulosti se BCI systémy navrhovaly na jednu danou činnost či úkon, a to z důvodu co největší přesnosti. Avšak rokem 2000 se situace změnila. Systém BCI2000 umožňoval využít více signálů s více výstupy.



Obrázek 10: Ukázka rozložení senzorů v systému EEG 10-20

2.3.1 Kanály EEG - Vědomé stavy

V rámci EEG odlišujeme několik typů signálů (vln) ve vědomých i nevědomých stavech. Každý typ se objevuje při určité interakci člověka a tyto signály jsou nejvíce rušeny.

Alpha (8-13Hz)

- Vědomí
- Vědomí těla, integrace pocitů
- Vyskytuje se s převahou nad okcipiálními částmi lbi během relaxované bdělosti a při zavřených očích. Po otevření očí dochází k rozpadu alfa aktivity a objevuje se aktivita rychlejší, ve frekvenčním pásmu Beta. Alfa vlny se objevují v částech mozku, kde se nalézají centra řeči.

Beta (14-30Hz)

- Myšlení
- Vnímání, koncentrace, mentální aktivita
- Beta aktivita se objevuje při zvýšení pozornosti po otevření očí, ale také při logickém a analytickém myšlení.

Gamma (30-64Hz)

- Vůle
- Extrémní soustředění
- Gamma aktivita se objevuje při extrémním soustředění, vypětí nebo stresu.

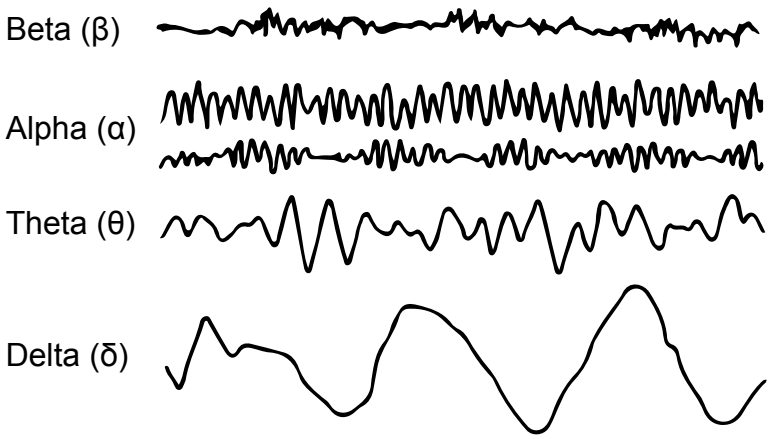
2.3.2 Kanály EEG - Nevědomé stavy

Theta (4-7,5Hz)

- Emoce
- Pocity, trans, sny
- Tyto vlny lze pozorovat u dětí do 13-ti let a u dospělých, kteří jsou ve stavu ospalosti nebo po probuzení. Objevují se i během meditace nebo při depresi.

Delta (0,5-4Hz)

- Instainkt
- Přežití, hluboký spánek, koma
- Pomalé frekvence Theta a stejně tak Delta se v bdělém stavu u zdravého dospělého jedince neobjevují. Fyziologicky jsou tyto aktivity přítomny během spánku, a nebo během vývoje jedince do 6. měsíce věku. V EEG se tyto vlny projevují jako vysoké amplitudy s nízkou frekvencí.



Obrázek 11: Ilustrace typů mozkových vln EEG signálu

3 Mozek

Tato diplomová práce si vyžadovala jisté základní znalosti anatomie mozku. Důvodem bylo zjistit, které části lidského mozku, využívané BCI systémy, je užitečné analyzovat. Následující kapitola popíše analyzované oblasti šedé kůry mozkové, které svou aktivitou umožňují systému BCI rozpoznávat to, na co člověk zrovna myslí.

Mozek, latinsky cerebrum, je dozajista nejužitečnější orgán lidského těla. Je řídícím a integračním orgánem nervové soustavy. Jeho primárním úkolem je řídit a kontrolovat veškeré tělesné funkce, jako je činnost srdce, trávení, řeči, pohyb, myšlení či vnímání emocí. Lidský mozek svou vahou 1300-1400g reprezentuje 2% celkové lidské hmotnosti. Obsahuje asi 50-100 miliard neuronů a až biliardu synaptických spojení. Anatomicky lze mozek rozdělit na několik základních částí, a to na prodlouženou míchu, mozeček, varolův most, střední mozek, mezimozek a koncový mozek. Následující kapitoly této práce se zaměří na koncový mozek, který je místem lidského vědomí.

Koncový mozek, latinsky telencephalon, je tvořen bílou kůrou mozkovou, kterou obaluje šedá kůra mozková. Koncový mozek je uspořádán do levé a pravé hemisféry. Koncový mozek tvoří 83% veškeré mozkové hmoty. Samotná mozková kůra je tvořena těly neuronů a neobsahuje nervové dráhy. V následující text se zaměří na konkrétní části koncového mozku, ve kterých vznikají podněty, akce a samotná lidská vůle. [3]

3.1 Významné části mozku

Jak již bylo výše řečeno, je koncový mozek tvořen několika oblastmi, které jsou významné svou funkcí a jsou to také jediné oblasti, kde lze sledovat aktivitu mozku bez zásahu do mozkové kůry či lebky. Nyní budou popsány tyto anatomicky a pro tuto práci významné části koncového mozku.

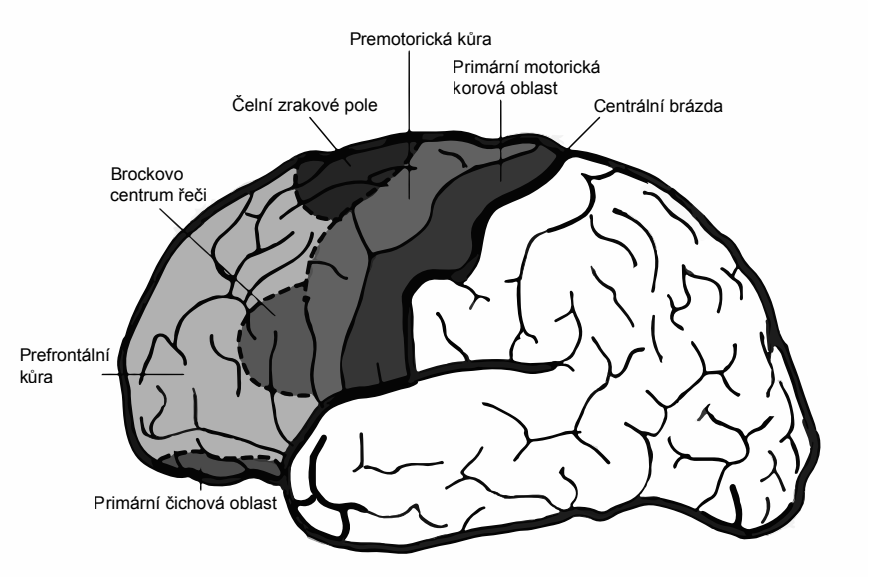
3.1.1 Čelní lalok

První zmíněnou částí je čelní lalok, nebo také frontální lalok (lobus-frontalis), který se dále anatomicky dělí na tři významné podoblasti, na které se tato práce zaměřuje. První podoblastí je motorický kortex (gyrus praecentralis) je závit mozkové kůry, ve kterém jsou uloženy neurony, vedoucí motorické impulzy do míchy. Tuto oblast nalezneme v Area 4 v Brodmannově mapě.

Druhou podoblastí frontálního laloku je motorické centrum řeči (gyrus frontalis inferior). Toto centrum je závit, v němž se nachází takzvané Brocovo centrum řeči, a je zodpovědné za řízení pohybů pro tvorbu mluvené i psané řeči. Tato oblast se nalézá v Area 44 v Brodmannově mapě.

Třetí významnou podoblastí je frontální asociační korová oblast (Area 46), integrující výstupy ze senzitivních sluchových, zrakových a rovnovážných oblastí mozku. Na základě těchto výstupů probíhá příprava motorické aktivity a jednání. Tato část se účastní na tvorbě kognitivních akcí, tedy vědomí, vnímání, vědomé zapamatování a rozpomínání, sebeuvědomění a sebekontroly. [3]

Je zjevné, že tyto oblasti budou klíčové, neboť jsou centrem kognitivních akcí a tyto akce budou, co se týče detekce, dominantní. Z toho důvodu je na tuto část mozku namapováno nejvíce senzorů headsetu EPOC.



Obrázek 12: Ilustrace oblasti čelního laloku, Zdroj: [5]

3.1.2 Temenní lalok

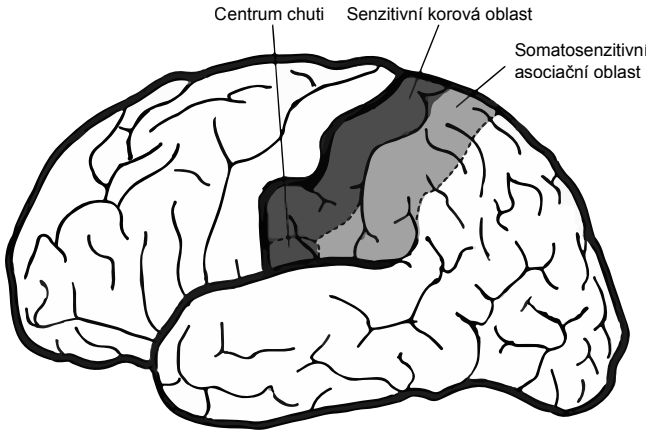
Druhou významnou částí koncového mozku je temenní lalok obsahující ty části mozku, analyzující vnější hmatové podněty. Tuto oblast můžeme rozdělit na dvě podoblasti. První je primární senzitivní korová oblast (gyrus postcentralis, Area 3,1,2), což je závit, v němž jsou uloženy cílové neurony, do kterých jsou vedeny senzitivní informace z těla. Tyto informace zde vstupují do vědomí. Příkladem jsou reakce na vnější podněty jako jsou chlad, teplo, vibrace, ale i hmat a bolest. V lékařství se pro vysvětlení této oblasti mozku využívá obrázku s názvem Homunkulus, jež ilustruje intenzitu vnímání těchto podnětů.

Druhou podoblastí temenního laloku je parietální asociační korová oblast (Area 7), jejíž funkcí je vyhodnocování a přiřazování významu senzitivním a zrakovým vjemům. V této části probíhá rozpoznávání tvarů a předmětů pomocí hmatu a přiřazení významů z paměti. [3]

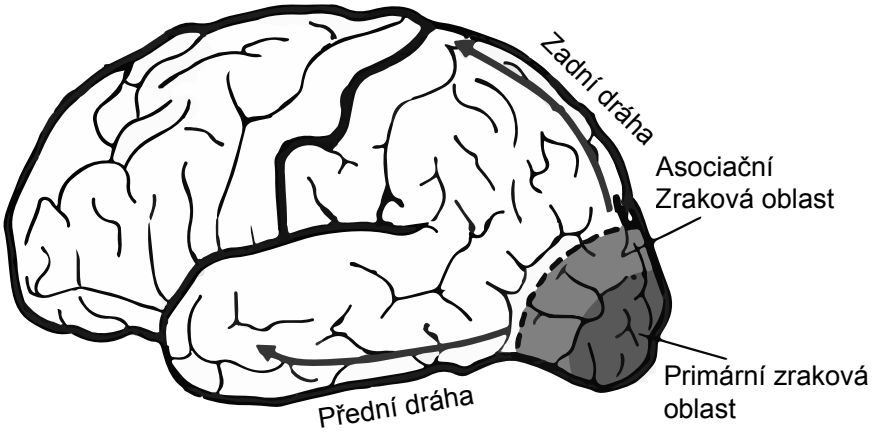
3.1.3 Týlní lalok

Třetí neméně významnou oblastí koncového mozku je týlní lalok, nalézající v zadní části mozku. Tento lalok je primární zrakovou oblastí, kde se nalézají poslední a cílové neurony zrakové dráhy. Mimo jiné zde také vstupují zrakové impulzy do lidského vědomí. V této oblasti je zvlášť významná okcipitální a asociační korová oblast (Area 17,18,19), která se

zabývá přiřazováním významu zrakového vjemu. Vznikají zde také souvislosti s jinými vjemy ve vědomí. Jinými slovy, vzniká zde asociace viděného objektu s objektem v paměti. [3]



Obrázek 13: Ilustrace oblasti temenního laloku, Zdroj: [5]

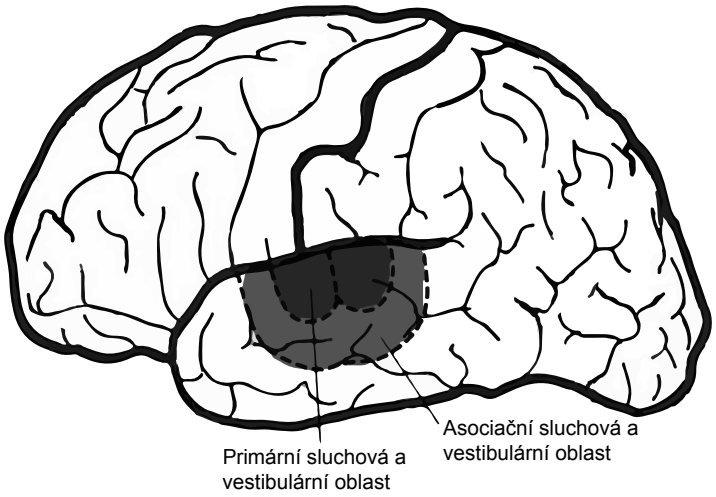


Obrázek 14: Ilustrace oblasti týlního laloku, Zdroj: [5]

3.1.4 Temporální lalok

Poslední částí mozku, významná pro tuto práci a celkově pro práci BCI, je temporální neboli spánkový lalok. Tato část mozku je sídlem centra sluchu a rovnováhy. Klíčovou podoblastí je primární sluchová a vestibulární korová oblast (Area 43), což je oblast

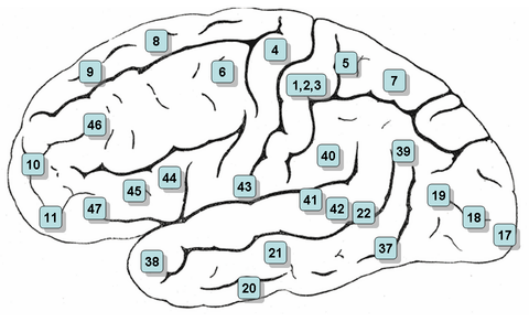
zodpovědná za vnímání sluchových impulzů. Tyto zvukové impulzy zde vstupují do vědomí. Kolem této oblasti je uložena Wernickeova oblast (Area 40), zodpovědná za rozpoznání a porozumění mluvené (slyšené) řeči. [3]



Obrázek 15: Ilustrace oblasti spánkového laloku, Zdroj: [5]

3.1.5 Brodmannova mapa

Brodmannova mapa je vizualizační nástroj umožňující viditelné rozčlenění mozku do oblastí s určitou schopností a funkcí. S tímto rozčleněním přišel německý vědec Korbinian Brodmann, kdy v roce 1909 předvedl první verze tohoto rozčlenění. Od té doby se tato mapa rozšiřovala a měnila, s nalezením nových oblastí až do dnešní podoby.



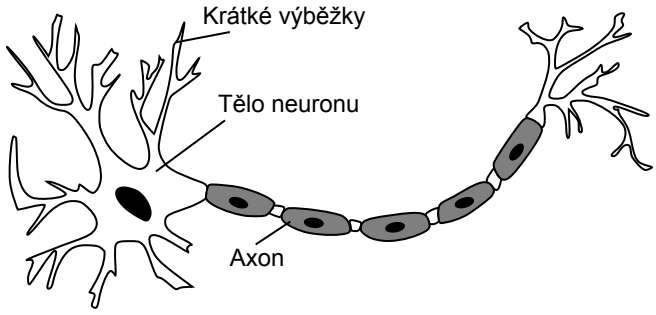
Obrázek 16: Brodmannova mapa

3.2 Neuron

Neuron je základní nervovou buňkou, tvořící nervovou tkáň. Neuron přenáší a následně zpracovává podněty vnitřního i vnějšího prostředí organismu. Neuron, a jeho vazby jsou zdrojem bioelektrické aktivity. Neuron je tvořen tělem neuronu, což je část nervové buňky, obsahující buněčné jádro a výběžky neuronu. Výběžky jsou spojnice, které spojují jednotlivé neurony, a umožňují jejich komunikaci. Neuron obsahuje dva typy výběžků. A to krátké, jež jsou součástí těla neuronu, a dlouhé, zvané Axony.

Existují čtyři druhy neuronů, a to unipolární, bipolární, pseudounipolární a multipolární.

- **Unipolární**, jsou neurony obsahující pouze jeden výběžek Axon. Příkladem unipolárního neuronu jsou neurony smyslové, jako jsou čichová buňka či tyčinky a čípky sítnice.
- **Bipolární**, jsou neurony, obsahující jeden dlouhý výběžek Axon a jeden výběžek krátký. Příkladem tohoto neuronu je druhý neuron zrakové dráhy.
- **Pseudounipolární**, jsou neurony zvláštního typu. Z těla neuronu vystupuje jediný výběžek, kombinující dlouhý a krátký výběžek. Tento výběžek se ale později dělí na dva. Příkladem pseudounipolárního neuronu jsou zakončení mozkových nervů.
- **Multipolární neurony**, jsou nejčastější neurony, kde z těla neuronů vystupuje několik krátkých výběžků a jeden dlouhý výběžek Axon.



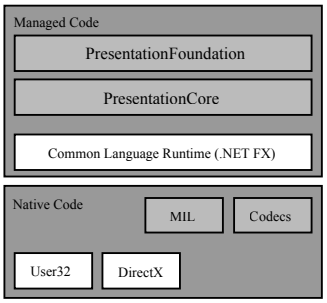
Obrázek 17: Vyobrazení buňky neuronu (Zdroj: http://www.clker.com)

4 Použité technologie

Následující kapitola popisuje klíčové frameworky a technologie, využívané v rámci této práce. Jsou zde zmíněny technologie, jako jsou Windows Presentation Foudnation, OpenGL a PhysX, které umožnily dosáhnout cílů této práce. Technologie WPF tvoří zásadní část všech aplikací, vytvořených v rámci této práce, jakožto technologie pro tvorbu grafického rozhraní (GUI). Tato technologie byla zvolena z důvodu jejího moderního přístupu ke tvorbě GUI a celkového návrhu aplikace. Jelikož využití výše zmíněných technologií bylo pro tuto práci stěžejní, budou jejich základní popisy zmíněny v následujících kapitolách.

4.1 Windows Presentation Foundation

Windows Presentation Foundation je část .NET Frameworku (3.0 a výše) určená pro tvorbu moderního uživatelského rozhraní. Oproti technologii WinForms přináší zcela nový přístup a typ definice návrhu uživatelského rozhraní. Největší změnou je jazyk XAML, vycházející z jazyka XML, rozšířený o některé další vlastnosti. Jednou ze základních vlastností WPF je vedení vývojáře k oddělení grafického návrhu od logiky aplikace. Grafický návrh je tvořen jazykem XAML a logickou částí kódu zvanou CodeBehind, která je reprezentována jazykem C# nebo VB.NET a od Windows 8 pak i c++.



Obrázek 18: Architektura technologie WPF

Vlastnosti WPF

- Jazyk XAML
- Využívá vektorovou grafiku
- DataBinding
- Stylování grafických prvků
- Animace grafických prvků pomocí Storyboard

- Akcelerace pomocí Direct3D
- Existence tříd pro základní 3D grafiku
- Možnost hostování WPF ve WinAPI a WinForms

4.1.1 Klíčové vlastnosti

Windows Presentation Foundation disponuje velkou spoustou novinek a zajímavostí. Asi nejvýraznější novinkou je již zmíněný jazyk XAML, který díky své syntaxi umožňuje navrhovat grafické prostředí aplikace velice rychle, efektivně a hlavně přehledně. XAML soubory lze zkompilovat do binárního souboru s koncovkou baml. Každý element jazyka XAML je reprezentací instance třídy a jednotlivé vlastnosti jsou reprezentovány atributy. Elementy lze vnořovat do sebe a ty pak tvoří takzvaný VisualTree. Díky VisualTree lze pak snadno přistupovat k jakémukoli elementu a jeho vlastnostem.

```
1  <Window x:Class="WPF_App.MainWindow"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4           Title="MainWindow" Height="350" Width="525">
5      <Grid>
6      </Grid>
7  </Window>
```

Výpis 1: Ukázka XAML

Další klíčovou vlastností, kterou WPF disponuje je DataBinding. O DataBindingu lze říct, že se jedná o svázání vlastností objektů. Typický příklad DataBindingu je navázání objektového modelu, například instance třídy Osoba, jež obsahuje vlastnost Name a Surname, na dva TextBoxy.

```
1  <StackPanel Orientation="Horitontal">
2      <TextBlock Text="Name:" />
3      <TextBox Text="{Binding Name}" />
4      <TextBlock Text="Surname:" />
5      <TextBox Text="{Binding Surname" />
6  </StackPanel>
```

Výpis 2: Ukázka DataBindingu

Klíčové slovo Binding říká, že do vlastnosti Text bude vázat vlastnost, například Name. Tato ukázka je však reprezentací jednocestného navázání ze strany zdroje, to jest, že změna vlastnosti Name se v TextBoxu projeví, ale změna hodnoty TextBoxu se ve vlastnosti Name nijak neprojeví. Toto je při použití TextBoxu nepřijatelné a proto lze Binding rozšířit o další nastavení. Jednou z klíčových vlastností Bindingu je Mode, který definuje vztah mezi oběma stranami Bindingu. Tuto vlastnost lze nastavit na OneWay (výchozí, jednocestné nastavení), TwoWay (dvou cestné svázání, kde příkladem užití je právě TextBox) a další speciální případy vazby. Poslední zmíněnou vlastností bude Convertor. Convertor umožňuje převádět vstupní data Bindingu na nějaká jiná výstupní data.

Typický příklad užití vlastnosti Converter je převod času nebo data do nějakého speciálního formátu. DataBinding tak umožňuje naplnit grafické prostředí daty, bez znalosti pojmenování grafických prvků. DataBindingu využívá pro XAML speciálně vytvořený návrhový vzor MVVM (Model-View-View-Model), který se snaží oddělit surová data od CodeBehind kódu.

Technologie WPF umožňuje upravovat vzhled aplikace do velmi velkých detailů. Lze tak snadno vytvářet vlastní ovládací prvky (komponenty), které mohou vypadat jakkoli. WPF nabízí také velkou škálu animací a efektů grafických prvků, které dodávají aplikacím dynamičnost. Možnost využít stylování, animace a mnoho dalších možností WPF, jsou velmi zajímavou ukázkou toho, jak je WPF moderní. Pomocí stylů lze prvky dynamicky měnit a to umožňuje další úroveň stylizace aplikace. Styl je objekt, popisující vzhled grafického prvků a skládá se z několika odvozených vlastností, které lze snadno rozšířit o vlastnosti vlastní. Jsou zde vlastnosti jako Background, Foreground popisující barvy pozadí a popředí, ale i klíčová vlastnost Template, která umožňuje popsat jakými grafickými prvky bude ovládací prvek tvořen. Další zajímavou a neméně důležitou vlastností jsou Triggry. Trigger je jakýsi rozhodovací člen, který umožní v definici stylu na základě podmínky nastavovat vlastnosti a chování. Existuje několik typů triggrů, umožňující rozhodování na základě hodnoty vlastnosti, hodnoty dat nebo například na základě vyvolané události.

Následující ukázka definice stylu tlačítka 3, vystihuje pouze základní možnosti stylování v XAMLu. Style obsahuje definici typu, kterého se týká, tedy v tomto případě třídy Button. První Setter nastaví výchozí hodnotu pro vlastnost Background. Druhý Setter nastavuje samotné složení komponenty. Konkrétně je zde tlačítko tvořeno pouze prvkem ContentPresenter (reprezentující obsah tlačítka), který je obklopen rámečkem realizovaný třídou Border. Border má nastavenou vlastnost Background, jejíž hodnotu převezme z výchozí nastavené hodnoty stylu pomocí TemplateBindingu, což je speciální typ Bindingu, dostupný pouze v definici stylu.

```
1  <Style TargetType="Button">
2    <Setter Property="Background" Value="Blue" />
3    <Setter Property="Template">
4      <Setter.Value>
5        <ControlTemplate TargetType="Button">
6          <Border x:Name="Border" Background="{TemplateBinding Background}">
7            <ContentPresenter />
8            </Border>
9          <ControlTemplate.Triggers>
10           <Trigger Property="IsMouseOver" Value="true">
11             <Setter TargetName="Border" Property="Background" Value="Black" />
12           </Trigger>
13         </ControlTemplate.Triggers>
14       </ControlTemplate>
15     </Setter.Value>
16   </Setter>
17 </Style>
```

Výpis 3: Ukázka Stylování třídy Button

Dále se zde nalézá definice Triggerů, obsahující jediný Trigger, který změní pozadí rámečku ve chvíli, kdy je vlastnost IsMouseOver nastavena na hodnotě True, tedy kdy je kurzor myši na tlačítku, a tím se docílí vizuální změny pozadí při najetí kurzoru na tlačítko.

Jak již bylo dříve řečeno, WPF nabízí zcela nový pohled na tvorbu grafického rozhraní a nabízí velkou spoustu nových možností. Za zmínku však stojí ještě jedna technologie vycházející z WPF. Tato technologie se nazývá Silverlight a jde o velmi odlehčenou verzi WPF pro použití ve webovém prohlížeči. Technologie Silverlight se aktuálně nachází ve své poslední verzi 5 a Microsoft tuto technologii pro svůj neúspěch dále nevyvíjí.

4.1.2 Základní ovládací prvky WPF

Základní třídou, ze které dědí všechny ovládací prvky, je FrameworkElement. Tato třída dědí z dalších tříd, ale lze ji považovat za základní element grafického rozhraní.

Prvky dědící z třídy Panel Prvky dědící z třídy Panel reprezentují komponenty sloužící jako kontejnery pro ostatní ovládací prvky.

- **Grid** – je základním panelem reprezentující layout mřížky. Definicemi RowDefinitions a ColumnDefinitions lze nastavit řádky a sloupce layoutu. Následující XAML kód ukazuje jak je snadné vytvořit mřížku 2x2 tlačítka, kde navíc první řádek bude mít výšku 50 pixelů a šířka obou sloupců bude dána 50:50 z celkové šířky mřížky.

```
1  <Grid>
2      <Grid.RowDefinitions>
3          <RowDefinition Height="50" />
4          <RowDefinition />
5      </Grid.RowDefinitions>
6      <Grid.ColumnDefinitions>
7          <ColumnDefinition />
8          <ColumnDefinition />
9      </Grid.ColumnDefinitions>
10     <Button />
11     <Button Grid.Column="1"/>
12     <Button Grid.Row="1" />
13     <Button Grid.Row="1" Grid.Column="1" />
14 </Grid>
```

Výpis 4: Ukázka definice panelu Grid

- **StackPanel** – je panel layoutu zásobníku. Jeho orientace může být vertikální nebo horizontální.
- **DockPanel** – panel umožňující dokovat vnořené prvky
- **Canvas** – panel umožňující kreslení
- **WrapPanel** – panel s automatickým layoutem

Prvky typu ContentControl ContentControl je základní třída, z níž dědí všechny třídy ovládacích a grafických prvků, které mohou obsahovat jeden další prvek. příkladem jsou následující prvky:

- **Window** - okno
- **Button** – obyčejné tlačítko
- **ToggleButton** – dvoustavové tlačítko
- **CheckBox** – zaškrtávací tlačítko

Prvky typu ItemsControl Prvky dědící z třídy ItemsControl, si lze představit jako seznamy. Typickými příklady využití jsou následující ovládací prvky:

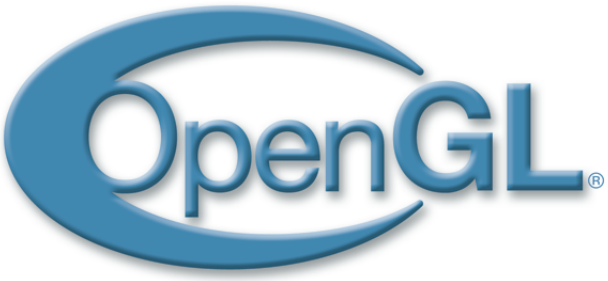
- **ListBox** – typický příklad seznamu
- **TabControl** – panel záložek
- **TreeView** – stromový seznam

Další typy ovládacích prvků Následující ovládací a grafické prvky vystihují prvky sloužící k zobrazování a zadávání informací.

- **TextBox** - standardní vstupní textové pole
- **TextBlock** - komponenta sloužící k zobrazování textových dat
- **Image** - komponenta zobrazující obrázky

4.2 OpenGL

OpenGL je softwarový standard popisující rozhraní grafické karty. Tento standard se skládá ze sady poměrně velkého počtu příkazů a vlastností, využívaných k vytváření hardwarově akcelerovaných 3D scén. Samotné OpenGL je navrženo tak, aby bylo multiplatformní, přenosné a hardwarově nezávislé rozhraní, což je jeho největší výhoda. Samotné OpenGL nemá implementovány žádné funkce, které by umožňovaly vytvářet okna a ošetřovat vstupy, z tohoto důvodu existují další rozšíření jako například Glut, které OpenGL obohacuje mimo jiné i o další schopnosti a vlastnosti. OpenGL pro svou jednoduchost implementuje pouze metody umožňující vytvářet objekty pomocí geometrických primitiv (bodů, úseček a mnohoúhelníků). Složitější popis objektů pak umožňuje rozšíření GLU (OpenGL Utility Library), kde lze například vytvářet a popisovat objekty pomocí Nurbs křivek a ploch.



Obrázek 19: Logo OpenGL (Zdroj: www.khronos.org)

4.2.1 Stručná historie

V osmdesátých letech dvacátého století se poprvé objevovaly grafické stanice umožňující zobrazovat krom textového režimu i základní grafiku. Tyto stanice disponovaly provozovat první grafickou knihovnu IrisGL vyvíjenou společností Silicon Graphichs Inc. IrisGL lze považovat za rodiče knihovny OpenGL. První verze OpenGL se objevila zhruba v roce 1992 a na rozdíl od IrisGL byla nezávislá na hardwaru a operačním systému.

- v1.0 - v1.5 (1992 - 2003)
- v2.0 - v2.1 (2004 - 2006)
- v3.0 - v3.3 (2008 - 2010)
- v4.0 - v4.4 (2010 - 2013)

Nyní se OpenGL spravováno Khronos Group

4.2.2 Klíčové vlastnosti OpenGL

Fixed Function Pipeline, dále jen FFP, je klíčovou vlastností a samotnou kostrou OpenGL. Tento řetězec umožňuje vytvořit, zpracovat a následně zobrazil scénu. FFP je tvořeno několika fixně danými a neměnnými stupni.

OpenGL, s příchodem verze 2.0 a grafických karet založených na GPU GeForce 3 a Radeon 8500, umožňuje využít mimo FFP také programovatelný řetězec. Programovatelný řetězec umožňuje řídit chování jednotlivých stupňů programem. Takovýto program se nazývá Shader. První shadery se programovaly nízkoúrovňovým programovacím jazykem nepodobným jazyku symbolických instrukcí. Tento jazyk byl však velice omezen. Později se objevil jazyk vyšší úrovně speciálně určený pro programování shaderů v OpenGL s názvem GLSL (OpenGL Shading Language). Aktuálně existují tři základní druhy shaderů a to Vertex, Fragment (pixel) a geometry shader.

Vertex shader je program, který se provede na každém vertexu geometrie scény. Příkladem využití vertex shaderu je například transformace vrcholu.

Fragment shader, neboli pixel shader, je druhou úrovní práce se shadery. Na rozdíl od vertex shaderu je program fragment shaderu prováděn na každém pixelu renderované scény. Pracuje s 2D obrazem scény a jeho nejčastějším užitím je aplikace a práce s texturami či modifikace barvy pixelu.

Geometry shader pracuje stejně jako vertex shader s vrcholy scény, ale na rozdíl od vertex shaderu umožňuje přidávat a odebírat nové vertexy. Geometry shader se využívá pro tvorbu objektů jako jsou například tráva.

Od verze OpenGL 3.2 je vykreslovací řetězec rozšířen o nové možnosti teselace. Přibyly tak dva nové shadery a to Tessellation control shader (TC) a Tessellation evaluation shader (TE). Tyto dva shadery se stejně jako geometry shader umožňuje dynamickou změnu scény.

4.3 PhysX

PhysX je framework aktuálně vyvíjený společností NVidia pro hardwarovou simulaci reálných fyzikálních zákonů. PhysX byl představen v roce 2004 společností Ageia a jeho hardwarová část byla implementována na samostatné PCI kartě. V roce 2008 PhysX koupila společnost NVidia, ta celou hardwarovou část integrovala do svých grafických karet. Technologie PhysX má podporu v mnoha platformách, mezi které patří Linux, Windows, MacOS X, ale i všechny moderní herní konzole.

Obrázek 20: Logo NVidia PhysX (Zdroj: www.nvidia.com)

Fyzikální model je tvořen několika základními typy objektů, kterým se říká aktéři. Prvním typem je statický aktér (Static Actor), reprezentující statický, v čase se neměnící a nepohybující se objekt, jako jsou stěny nebo pevné překážky. Díky tomu, že se statický aktér nemění v čase, je jeho fyzikální výpočet nejméně náročný. Dalším typem je aktér kinematický (Kinematic Actor), který se od statického liší v tom, že se může ve scéně pohybovat, ale nebude ovlivněn jinými objekty. Příkladem kinematického aktéra jsou animované objekty, jejíž pohyb je předem dán. Posledním typem aktéra ve fyzikální scéně je dynamický aktér. Tento typ, jako jediný, se může ve scéně volně pohybovat, a je možné ho ovládat. S větším množstvím dynamických aktérů roste nárok na výpočty fyzikálního modelu scény. Příkladem dynamického aktéra je například herní postava nebo kulička v bludišti.

Fyzikální scéna může být ve PhysX vytvořena buďto z primitiv, které jsou součástí SDK, nebo lze scénu načíst z modelu. Pro vytvoření scény z modelu je nutné mít seznam vertexů, a jejích indexů, lze tedy použít stejné datové struktury jako v OpenGL.

```
1   float* vertices = &vertexIndex[ii].Vertices[0];
2   int* indices = &vertexIndex[ii].Indices[0];
3   PxTriangleMeshDesc p;
4   p.points.count = vertexIndex[ii].Vertices.size()/3;
5   p.points.stride = 4*3;
6   p.points.data = vertices;
7   p.triangles.count = vertexIndex[ii].Indices.size()/3;
8   p.triangles.stride = 4*3;
9   p.triangles.data = indices;
10  PxToolkit::MemoryOutputStream writeBuffer;
11  PxCookingParams cookingP;
12  cookingP.suppressTriangleMeshRemapTable = true;
13  cookingP.buildTriangleAdjacencies = true;
14  PxCooking *cooking = PxCreateCooking(PX_PHYSICS_VERSION, *mFoundation,
        PxCookingParams());
15  cooking->setParams(cookingP);
16  bool status = cooking->cookTriangleMesh(p, writeBuffer);
17  if (!status) return;
18  cooking->release();
19  PxToolkit::MemoryInputData readBuffer(writeBuffer.getData(), writeBuffer.getSize());
20  PxTriangleMesh *tmesh = gPhysicsSDK->createTriangleMesh(readBuffer);
21  PxRigidStatic* meshActor = gPhysicsSDK->createRigidStatic(PxTransform(PxVec3(0.0,0.0,0.0)
        ,qa1));
22  PxShape* meshShape;
23  PxTriangleMeshGeometry triGeom(tmesh);
24  meshShape = meshActor->createShape(triGeom, *mMaterial);
```

Výpis 5: Ukázka vytvoření části fyzikálního modelu z načteného modelu

Kód 5 ukazuje vytvožení fyzikálního modelu podle 3D modelu. Model je popsán bufferem vertexů a indexů. Vytvoří se instance třídy PxTriangleMesh, která se využije v takzvaném Cookingu, tedy v postupu, kdy se model popsaný vertexy převede na fyzikální objekty, se kterými umí PhysX pracovat.

5 Aplikace

Úkolem této práce bylo vytvořit sadu aplikací, obsluhující a spravující náhlavní soupravu EPOC od společnosti Emotiv. Dalším úkolem bylo vytvořit vizualizaci mozkové aktivity, reprezentovanou jako komponentu integrovanou do jedné z aplikací. Vznikly celkem tři aplikace, z nichž jedna slouží jako správce headsetu EPOC a dvě jako aplikace doplňkové.

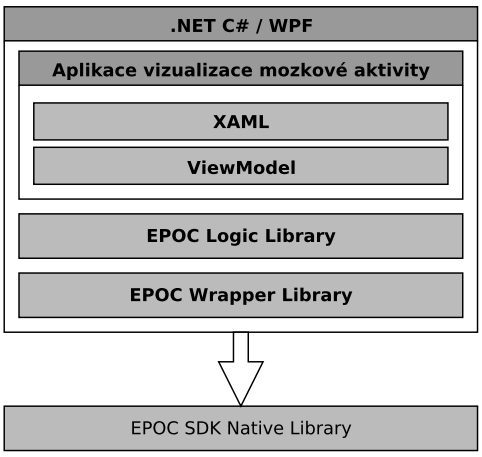
První zmiňovanou aplikací bude aplikace jejíž úkolem je spravovat headset EPOC, a obsahuje komponentu, která umožňuje vizualizace mozkové aktivity, která je tímto headsetem snímaná.

Druhou zmiňovanou aplikací bude hra, reprezentující příklad použití BCI systému v reálné aplikaci. Záměrem této hry, bylo umožnit představu, jak vlastně BCI systém funguje a k jakým účelům jej lze využít. Hru tvoří scéna složená z hrací plochy, realizované nakloněnou rovinou v níž jsou rozmístěny překážky a padající kuličkou, kterou se snaží subjekt, ovládající hru, zachytit.

Třetí aplikace slouží jako návrhář scény hry. Aplikace umožňuje vkládat a spravovat objekty na hrací ploše.

5.1 Aplikace pro správu headsetu EPOC

Úkolem této práce bylo vytvořit aplikací umožňující správu headsetu Emotiv EPOC a vizualizaci mozkové aktivity snímané headsetem EPOC. Aplikace se skládá ze tří vrstev, a to vrstvy obsahující grafické rozhraní aplikace, vrstvy obslužné logiky nativní knihovny a v poslední řadě, vrstvu, kterou tvoří nativní knihovna. Nativní knihovna je spolu s dalšími pomocnými knihovnami jako je wrapper, součástí Emotiv EPOC SDK. Celá struktura aplikace je vidět na obrázku 21.



Obrázek 21: Struktura aplikace pro správu headsetu

Nejnižší vrstvu tvoří nativní knihovna, která je součástí Emotiv EPOC SDK. Jedná se o aplikační jádro napsané v jazyce c++ a implementující rozhraní pro externí přístup. Tato knihovna je zároveň rozhraním propojující hardware s aplikačním jádrem.

Mezivrstvou mezi nativní knihovnou a grafickou částí aplikace jsou dvě .NET knihovny. První knihovnou mezivrstvy je knihovna wrapperu EmotivSDK.dll, tvořící rozhraní mezi nativní knihovnou a knihovnami napsanými v .NET frameworku. Tato knihovna tedy neobsahuje žádnou složitou logiku, obsahuje pouze definice tříd a metod, které následně volají metody z nativní knihovny. Tuto knihovnu využívá knihovna EPOC_EEG_Library.dll, na obrázku 21 nazvaná EPOC Logic Library. EPOC Logic Library implementuje sadu metod a událostí, spravujících logiku potřebnou pro ovládání nativní knihovny a tudíž samotného headsetu. Základním jádrem této knihovny je ve vlastním vlákně běžící proces, kontrolující stav headsetu. Tento proces je realizován nekonečnou smyčkou, která, aby zbytečně nevytěžovala procesor, je pozastavována krátkým intervalem. S ohledem na to, že v této smyčce dochází mimo jiné i ke sběru dat, musel být tento interval zvolen vhodně tak, aby nedošlo k zbytečnému úniku informací. V rámci této nekonečné smyčky se opětovně volá metoda ProcessEvents hlavní třídy SDK EmoEngine. Metoda ProcessEvents má za úkol kontrolovat stav Headsetu, a pokud došlo k nějaké změně v logice nativní knihovny, pak třída EmoEngine vyvolá patřičnou událost, která se zachytí a zpracuje. Knihovna EPOC Logic Library také obsahuje metody pro správu komunikace headsetu a komunikace s externími aplikacemi pomocí UDP paketů, dále se zde nalézají metody pro správu a učení kognitivních metod.

```
1  void DoWork()
2  {
3      long lastTics_Update = 0;
4      long lastFail_Time = System.DateTime.Now.Ticks;
5      long dataRate_updateTime = 1500000;
6      while (Collect)
7      {
8          try
9          {
10             long sleep = dataRate_updateTime - (System.DateTime.Now.Ticks - lastTics_Update);
11             if (System.DateTime.Now.Ticks - dataRate_updateTime < lastTics_Update)
12                 if (sleep > 0 && sleep < dataRate_updateTime * 2)
13                     System.Threading.Thread.Sleep(50);
14             lastTics_Update = System.DateTime.Now.Ticks;
15             Action();
16 ...
17 void Action()
18 {
19     engine.ProcessEvents(300);
20     if ((int)User == -1) return;
21     data = engine.GetData((uint)User);
22     if (data == null) return;
23     BufferSize = data[EdkDll.EE_DataChannel_t.TIMESTAMP].Length;
24     Data2 = new Dictionary<string, double>();
25     foreach (EdkDll.EE_DataChannel_t channel in data.Keys)
26     {
27         double[] tmp = data[channel];
28         Data2.Add(channel.ToString(), tmp[0] - tmp.Sum() / tmp.Length);
```

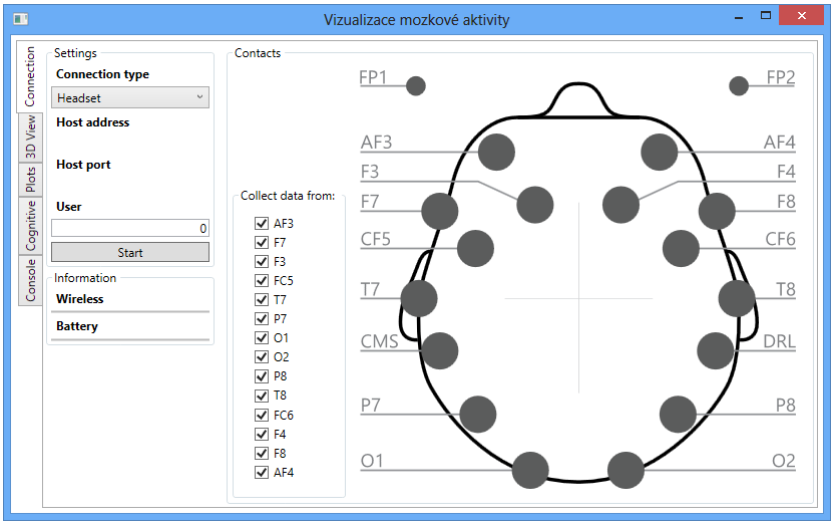
Výpis 6: Část obslužného kódu pro získávání dat

Výše uvedený kód reprezentuje jádro knihovny Logic Library. Metoda DoWork běží ve vlastním vlákně a jejím úkolem je obsloužit metodu Action. Vzhledem k tomu, že se metoda Action volá v relativně nekonečném cyklu, bylo nutné z důvodu výkonu zavést opatření. Opatřením se myslí přerušování/pozastavování cyklu na 50ms. Metoda Action pak volá již zmiňovanou metodu ProcessEvents z třídy EmoEngine, dále se zde kontroluje zda existuje aktivní uživatel. Aktivním uživatelem se myslí připojené zařízení EPOC. Pokud existuje nějaký uživatel, získají se data a vytvoří se struktura tvořena klíčem, což je název kontaktu a patřičnými daty. Následně se vyvolává událost oznamující změnu dat, která se propaguje dál do aplikace.

Poslední vrstvou je samotná aplikace s grafickým rozhraním. U .NET aplikací existují dvě základní technologie, umožňující vytvářet grafické rozhraní aplikací. První technologií je WinForms, reprezentující základní a hojně využívanou technologii. Druhou a modernější technologií je WPF, která umožňuje větší možnosti úprav vzhledu než WinForms. Pro tuto aplikaci byla zvolena technologie WPF. Aplikace je graficky rozdělena do několika sekcí, které spravují konektivitu headsetu, zobrazují data, spravují kognitivní akce a vizualizují samotnou aktivitu mozku.

5.1.1 Connection

První karta "Connection"spravuje konektivitu headsetu nebo nástrojů, které jsou součásti Emotiv SDK. Aplikaci tak lze připojit k headsetu EPOC nebo aplikaci EmoComposer, umožňující virtualizovat headset. Karta dále umožňuje volbu uživatele a vizualizuje stav baterie a síly signálu. Nalézá se zde grafické znázornění rozložení kontaktů headsetu, kde je každý kontakt reprezentován puntíkem, který mění svou barvu na základě kvality signálu. A v neposlední řadě se zde nalézá seznam, umožňující výběr kontaktů, u kterých bude docházet ke sběru dat.

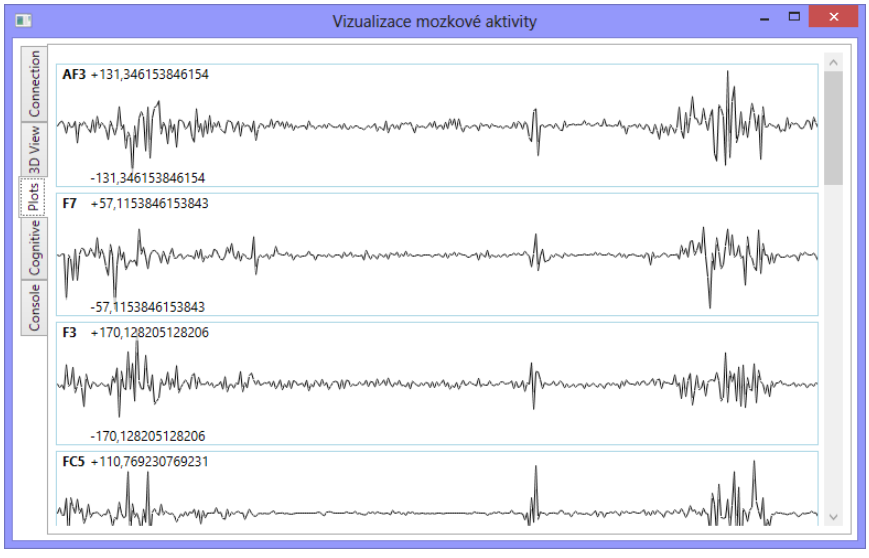


Obrázek 22: Záložka sloužící ke správě headsetu

- **Černá** - bez signálu
- **Červená** - velice špatná kvalita signálu
- **Oranžová** - špatná kvalita signálu
- **Žlutá** - dobrá kvalita signálu
- **Zelená** - velmi dobrá kvalita signálu

5.1.2 Plots

Další sekcí(záložkou) je "Plots", která má za úkol graficky vizualizovat signály získané headsetem v grafech, kde každý graf reprezentuje signál daného kontaktu. Požadavek na tuto část byl jasný a to zobrazovat čtrnáct grafů v reálném čase. To se zprvu zdálo jako snadný úkol, ale později se projevily některá omezení technologie WPF. Hlavním problémem bylo efektivně vykreslit několik set až tisíc bodů spojených čarou v jednom grafu. Vyzkoušelo se několik knihoven umožňujících vykreslování grafů, ale ani jedna nedodávala dostatečný výkon a navíc velmi vytěžovaly procesor. S ohledem na poměrně velké nároky na vykreslování, bylo tedy nutné pro účely vykreslování grafů vytvořit vlastní grafický prvek. Základem se stala technologie GDI, která je dostatečně rychlá a poskytuje, jak se později během implementace ukázalo, žádané výsledky. GDI je poměrně zajímavá technologie, jež je jednou ze tří základních částí systému Windows. Úkolem GDI je rychle a efektivně vykreslovat body, čáry, křivky a fonty. Z tohoto důvodu GDI běží hardwarově. Graf je reprezentován třídou PlotItem, starající se o vykreslování grafu. Na pozadí každého grafu je struktura nesoucí vzorek dat. Tento vzorek dat je uložen v cyklickém Bufferu jehož základ tvoří obyčejné pole a plní se podobně jako zásobník.



Obrázek 23: Vykreslování grafů pomocí technologie GDI

Tento postup bylo potřeba vylepšit, a to z důvodu častého posunu hodnot v bufferu, což je časově i paměťově poměrně náročné. Řešení znamenalo přidat dva indexy, jejichž úkol je ukazovat na stav bufferu. První reprezentuje ukazatele aktuálního stavu bufferu a druhý slouží jako ukazatel pozice posledního přidaného záznamu. Druhý index má zvláštní funkci. Jakmile je buffer poprvé naplněn, přidává se každá nová hodnota na pozici druhého pomocného indexu, ten se poté posune o jednu pozici doprava. Jakmile je druhý pomocný index na konci bufferu, přesune se opět na začátek. Získání dat se pak provádí vybíráním hodnot od pozice druhého pomocného indexu po konec bufferu a následně od začátku bufferu pro pozici druhého indexu.

Dalším problémem se ukázal fakt, že je nutné, aby každý graf běžel ve vlastním vlákně. Pokud všechny grafy běžely v hlavním vlákně GUI, docházelo k nepříjemnému zasekávání aplikace. S ohledem na to že ve WPF teoreticky existuje pro GUI pouze jedno vlákno, bylo nutné vymyslet způsob jak toto omezení obejít. Problém vykreslování grafu a problém výkonu celé aplikace zapříčinil vznik grafické komponenty Plot, starající se o vytváření všech čtrnácti grafů a jejich obsluhu.

Následující část kódu ukazuje vytvoření instance grafického prvku pro vykreslování grafu ve vlastním vlákně. Základem všeho je pomocná třída VisualWrapper, do níž se vkládá ve svém vlákně vytvořená instance třídy HostVisual nesoucí samotný grafický prvek grafu.

```
1  private HostVisual CreateVwThread()
2  {
3      HostVisual hostVisual = new HostVisual();
4      Thread thread = new Thread(new ParameterizedThreadStart(VwThread));
5      thread.ApartmentState = ApartmentState.STA;
6      thread.Start(hostVisual);
7      s_event.WaitOne();
8      return hostVisual;
9  }
10 int t = 3;
11 private FrameworkElement CreateVw()
12 {
13     PlotItem2 pi = new PlotItem2();
14     Items.Add(Globals.targetChannelListNames[t++], pi);
15     return pi;
16 }
17 private void VwThread(object arg)
18 {
19     HostVisual hostVisual = (HostVisual)arg;
20     VisualTargetPresentationSource visualTargetPS = new VisualTargetPresentationSource(
           hostVisual);
21     s_event.Set();
22     visualTargetPS.RootVisual = CreateVw();
23     System.Windows.Threading.Dispatcher.Run();
24 }
25  ...
26 VisualWrapper vw = new VisualWrapper();
27 vw.Child = CreateVwThread();
```

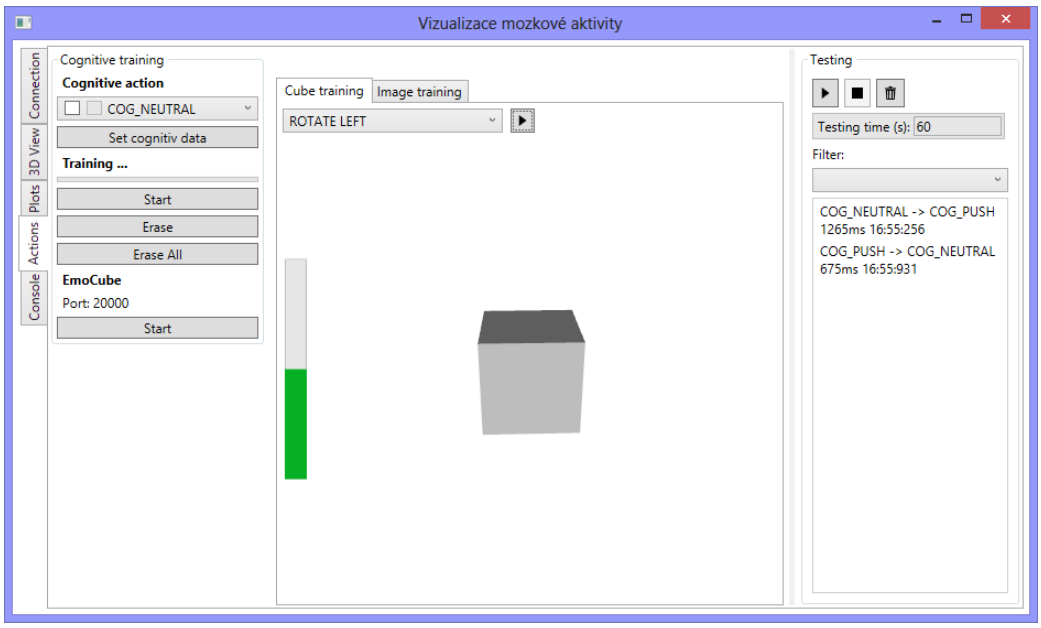
Výpis 7: Vytvoření instance ovládacího prvku Plot ve vlastní vlákně

Vzhledem k velmi časté změně intenzit signálů bylo dále nutné zařídit jistou přizpůsobivost grafu signálu. Byl zvolen postup, který realizoval změnu měřítka osy Y v závislosti na získaném signálu.

V průběhu signálu se v bufferu, obsahujícího aktuální úsek dat, detekovalo globální maximum hodnot signálu. Každá nová hodnota se porovnává s již nalezeným globálním maximem bufferu. Takto bylo možné rychle reagovat na nově příchozí změnu globálního maxima. Bylo však nutné také ošetřit stav, kdy globální maximum bude nahrazeno novým globálním maximem, obsaženým někde v bufferu. Tato situace nastane ve chvíli kdy zjištěné globální maximum se už v bufferu nenalézá. Buffer se tedy prochází a hledá se nové globální maximum.

5.1.3 Actions

Další záložka s názvem "Actions"reprezentuje nástroje pro správu kognitivních akcí a jejich testování. Záložka je vizuálně rozdělena na ovládací panel obsahující vizualizaci trénovací kostky a trénovacích obrázků. Ovládací panel pak poskytuje seznam definovaných kognitivních akcí. V tomto seznamu je vizuálně vidět, které akce jsou již naučené a lze v něm vybírat akce, které mají být v dané chvíli systémem detekovatelné. Naučení vybrané akce se inicializuje kliknutím na tlačítko "Start"v sekci "Training". Naučené akce lze mazat buďto všechny najednou nebo lze vymazat pouze aktuálně vybranou akci. Uprostřed okna se pak nalézá sekce, která pomáhá v trénování kognitivních akcí. Jsou zde dvě záložky, z nichž první "Cube training"umožňuje vizualizaci pohybů krychle. Druhá záložka nazvaná "Image training"má za úkol usnadnit trénování akcí pomocí vizuálního podnětu statického obrázku.



Obrázek 24: Správa kognitivních akcí a testování

Aplikace prohledává složku ”TrainImages”ve složce aplikace, kde hledá obrázky a umožňuje jejich zobrazování. Podporované soubory jsou obrázky typu PNG a JPEG. V levém horním rohu se nalézá ovládání automatického promítání těchto obrázků. Lze zde nastavit interval, reprezentující čas, jak dlouho bude obrázek zobrazen než se zobrazí obrázek další. Dole se pak nalézají tlačítka pro ruční přepínání obrázků. V pravé sekci okna se nalézá panel umožňující záznam detekovaných kognitivních akcí. V tomto panelu lze nastavit interval trvání, po kterou se budou detekované akce vypisovat. Testování se spustí kliknutím na tlačítko ”Play”a zastaví kliknutím na ”Stop”. Tlačítko popelnice slouží pro smazání seznamu, který se nalézá níže. V seznamu výsledků lze filtrovat podle detekované akce. K filtraci slouží výběr, který se nachází nad seznamem. V tomto výběru se nalézá seznam naučených akcí.

Záznam ze seznamu zachycených akcích poskytuje informace o akcích, které se vyvolaly a interval změny.

5.1.4 3D View

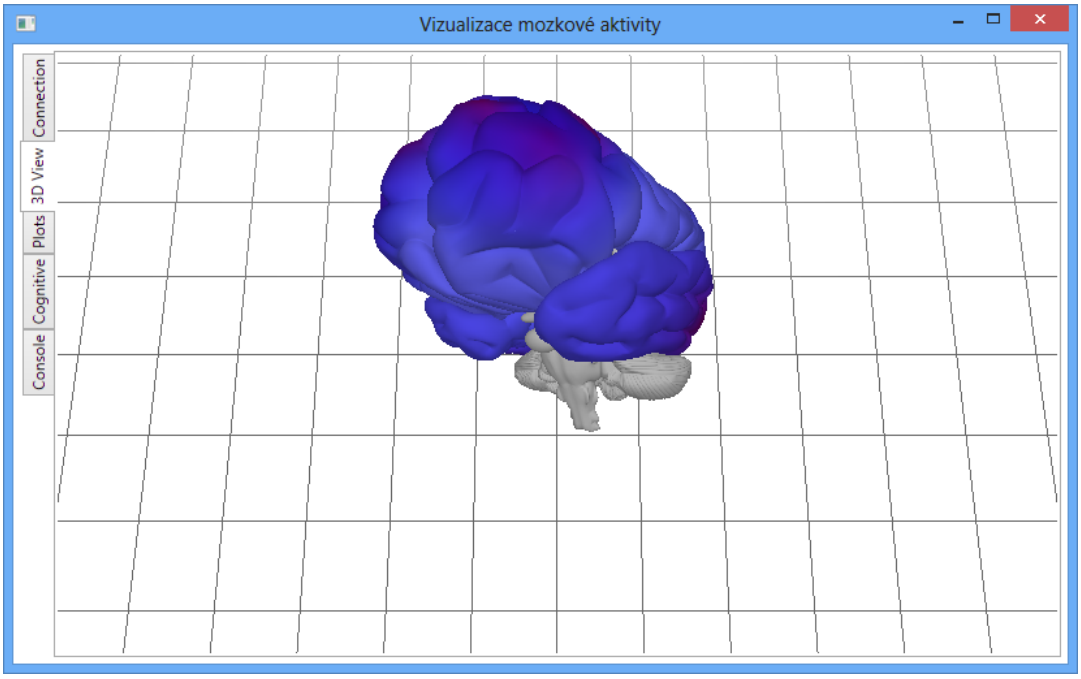
Záložka s názvem ”3D View”obsahuje samotnou vizualizaci mozkové aktivity. Základní myšlenkou bylo zobrazit na modelu mozku aktivitu kolem předem určených bodů. Zobrazování modelu a samotné vizualizace je realizována samostatnou knihovnou jazyka C++ CLI/CLR (tzv. Managed C++) spolu s technologií OpenGL. Technologie CLI/CLR je zde využita z důvodu integrace OpenGL do hlavní aplikace, která je ovšem napsána v jazyce C#. CLI/CLR jsou knihovny napsané v jazyce c++ s možností využití technologie .NET frameworku, díky tomu bylo možné v této knihovně využít OpenGL a přitom ji bez problému napojit na primární aplikaci.

Aby bylo možné ve WPF zobrazovat obraz vytvořený v OpenGL, bylo nutné vytvořit grafickou komponentu, která by OpenGL hostovala. Tato komponenta je tvořena třemi vrstvami, skrz které se volají pomocné metody přenášející data, reprezentující intenzity signálu mozkové aktivity.

První vrstva je tvořena uživatelským prvkem WPFOpenGLControl obsahujícím komponentu WindowsFormsHost, která hostuje druhou vrstvu. Komponenta WPFOpenGLControl je čistý WPF/C# uživatelský prvek, který mimo zobrazování také zachycuje a částečně zpracovává akce myši, jako jsou rotace a zoom.

Druhá a třetí vrstva je obsažena v již zmiňované CLI/CLR knihovně. Druhá vrstva vznikla z důvodu oddělení čistého C++ a CLI a stará se o fyzické vykreslení obrazu z OpenGL a přeposílá data z WPF do OpenGL.

Třetí vrstva je reprezentací samotné vizuální stránky vizualizace. Je napsána v čistém C++ a využívá aplikační rozšíření OpenGL. Poměrně složité bylo získat nějaký pěkně vypadající model lidského mozku. Na internetu existuje poměrně dost webových stránek poskytujících velmi věrohodné modely, ale všechny tyto modely byly poměrně drahé. Po dlouhém hledání se podařilo nalézt model, který byl zadarmo. Na tomto modelu bylo však nutné provést několik úprav a změn. První změnou bylo spojení meshů do 3 větších celků. Dále se musel model, z důvodu poměrně velké hrubosti, zjemnit. Všechny tyto změny vydaly celkem pěkný výsledek. Jakmile byl získán použitelný a docela pěkně vypadající model, bylo nutné se zamyslet, jakým způsobem provést mapování a zvýra-



Obrázek 25: Vykreslování grafů pomocí technologie GDI

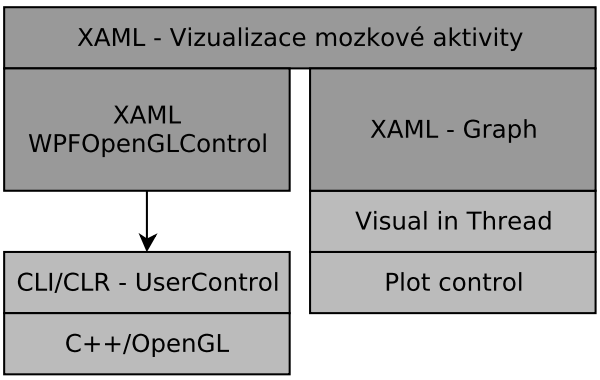
zňování těch oblastí modelu mozku, které jsou aktivní. Byl tedy zvolen postup, kdy se samotné vykreslování oblastí a intenzity aktivity, převede do shaderu, tedy programu běžícího na straně grafické karty. Bylo potřebné nalézt postup, který by umožnil přiřadit textuře pozice patřičných kontaktů. K tomuto účelu posloužil program Blender, který byl také použit ke spojování a zjemňování modelu. Blender umožnil vygenerovat potřebné texturovací souřadnice a také vytvořit pomocnou texturu. Tato pomocná textura se Blenderu nanesla na model a následně se do této textury zanesly body, odpovídající pozicím senzorů headsetu. Textura se následně uložila a byla použita pro získání souřadnic senzorů v textuře. Takto získané souřadnice se pak využily k určení vykreslovaných oblastí ve Fragment shaderu. Do shaderu jsou přenášena data reprezentovaná polem o 14-ti prvcích, obsahující intenzitu všech senzorů. V shaderu se pak počítá jednoduchým výpočtem radius a gradient kolem každého předem získaného bodu reprezentujícího souřadnice senzoru.

```
1  power = params[11];
2  //souradnice x a y reprezentuji texturovaci souradnici pozice senzoru
3  x = 0.1933;
4  y = 0.2578;
5
6  x1 = x -vTexCoord.x;
7  y1 = y -vTexCoord.y;
8  radius = 0.05+power*0.2;
9  startColor = 1.0;
10 coef = 1.0;
```

```
11  coef = 0.5+power*0.4;
12  m = (x1)*(x1)+(y1)*(y1)-(radius*radius);
13
14  if (m < 0)
15  {
16      float cco = abs(distance(vec2(x, y), vTexCoord.xy)/radius);
17      vec4 t = texture1D(myTexture, startColor+(1.0f-cco-0.01)*coef).rgba;
18      vec4 b1 = mix(vec4(t.r, t.g, t.b, 1.0f), finalColor, cco);
19      finalColor = mix(b1, finalColor, cco);
20      finalColor = b1;
21  }
```

Výpis 8: Část fragment shaderu

Část kódu fragment shaderu 8 reprezentuje zmiňovanou logiku výpočtu oblasti. Pole params obsahuje intenzity získané z headsetu, z tohoto pole se v tom případě vybere 11-tý záznam. Jsou zde vidět získané souřadnice, konkrétně 11-tého senzoru a následný výpočet parametru radiusu, ten se porovnává s 0, pokud je parametr menší než nula, bude se vykreslovat. Pokud je podmínka splněna, počítá se hodnota barvy v daném vykreslovaném pixelu, v závislosti na pozici od středu a na hodnotě vzorku gradientu. Výsledná barva se mixuje buďto s výchozí barvou nebo s barvou gradientu, který zasáhl do stejné oblasti. Výsledná barva mixování je následně podrobena výpočtu Phongova osvětlovacího modelu.



Obrázek 26: Diagram struktury externích ovládacích prvků

5.2 Hra

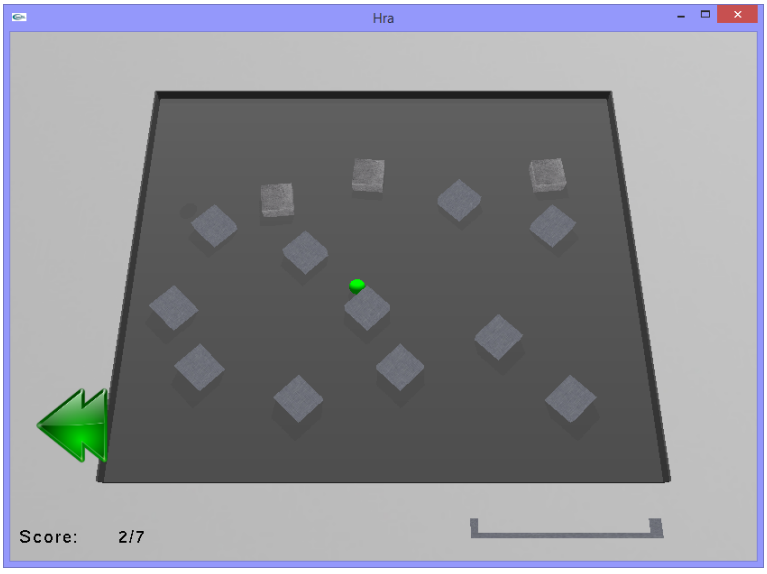
Aplikace hry byla vytvořena jako prezentace použitelnosti headsetu. Komunikace mezi touto aplikací a aplikací spravující headset je řešena pomocí UDP packetů. S ohledem na obtížnost ovládání čehokoli pouhým myšlením, byla aplikace navržena tak, aby její ovládání bylo co nejsnazší. Princip hry je jednoduchý. Hrací plochou je nakloněná rovina s překážkami, jako jsou krychle nebo jiné objekty. V horní části se nalézá oblast, kde

se generuje kulička, která padá po nakloněné rovině dolu a odráží se od překážek. Ve spodní části pod hrací plochou se nalézá zachytávací objekt. Aplikace se ovládá buďto klávesnicí (šipkami vpravo a vlevo), nebo naučenými akcemi z primární aplikace (Push a Pull). Může nastat situace, kdy se padající kulička zasekne v hrací ploše. Pro tuto situaci existuje funkce vygenerování nové kuličky, která se vyvolává klávesou šipky dolů.

Aplikace je napsána v jazyce c++ a za pomocí aplikačního rozhraní OpenGL a NVidia PhysX. OpenGL se stará o vykreslování scény, zatímco PhysX je zde použit za účelem chování a dynamičnosti scény. PhysX tedy vytváří jednoduchý fyzikální model scény, odpovídající rozmístění objektů načteného z datového souboru. Datový soubor je vytvořen buďto ručně, podle přesně určených pravidel, nebo je vygenerován za pomocí speciální aplikace s názvem Generátor scény. Start aplikace provází sekvence úkonů: Načtení datového souboru a jeho následné rozparsování - načtení potřebných modelů - vytvoření scény - start UDP serveru - vykreslování scény. Navázání komunikace mezi hrou a primární aplikací vyžaduje nejprve start hry a následně i primární aplikace. V primární aplikaci se naváže spojení s headsetem EPOC a naučí se potřební kognitivní akce. Samotné spojení se inicializuje kliknutím na tlačítko "Start"v sekci EmoCube.

Načítání modelů scény bylo realizováno pomocí Assimp Importeru, který umožňuje vytvářet jednoduchou datovou strukturu modelů. Takto získané vertexy, normály, materiály a textury bylo nutné převést do speciální struktury, kterou aplikace využívá. Dalším krokem bylo dopočítání indexů vertexů, které byly potřebné pro vytvoření fyzikálního modelu. Během načítání modelů probíhá výpočet jejich pozice podle předem určených pozicí, načtených z datového souboru.

Následně probíhá Vytvoření fyzikálního modelu pomocí technologie PhysX a vykreslování scény technologií OpenGL. Během vykreslování se mimo jiné vykreslují i jemné stíny, vrhané na hrací plochu objekty scény.



Obrázek 27: Ukázka aplikace Hra

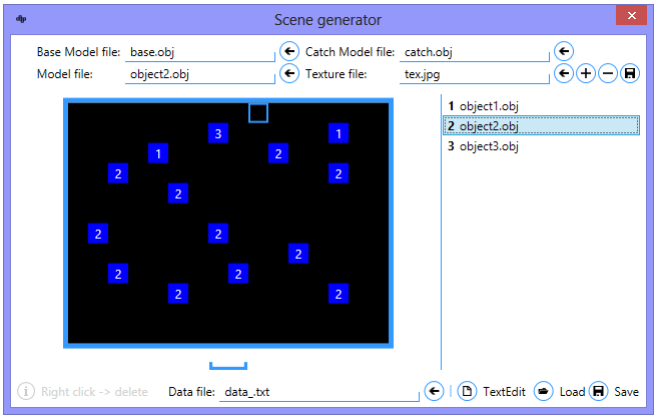
5.2.1 Generátor scény

Generátor scény je doplňková aplikace vytvořená technologií WPF, umožňující navrhovat hrací scénu pro aplikaci Hra. Tato aplikace umožňuje přidávat do scény nové objekty, které jsou definovány modelem (soubor Wavefront .obj) a obrázkem textury. Dále se zde definuje zachytávací objekt a model hrací plochy. Seznam objektů v panelu na pravé straně aplikace umožňuje jejich vkládání na hrací plochu. Kliknutím levého tlačítka na hrací plochu se objekt vloží na zvolené místo a kliknutím pravým tlačítkem na objekt v hrací ploše se objekt smaže. Aplikace umožňuje scénu uložit, načíst a upravovat. Datový soubor je pak jednoduchý textový soubor, který je tvořen přesně danou strukturou. Ukázka struktury se nalézá v příloze A.

Definice struktury datového souboru scény

- **size** definuje velikost hrací plochy. Doporučená velikost je 16x12
- **objects** reprezentuje seznam objektu ve scéně. Každý řádek definuje jeden objekt a jeho definice má pevný formát *Identifikátor*;*cesta k obj modelu*
- **textures** je seznam textur objektů. Formát je opět pevně identifikátorem, který odkazuje na objekt a cestou k obrázku textury.
- **data** tato část obsahuje počet řádků a sloupců podle definice velikosti ze sekce size, tedy obsahuje 16x12 znaků, kde znak je identifikátor objektu.

Identifikátory objektů se pak dělí do dvou skupin. První skupina dána identifikátorem -1 a 0, kde -1 je objekt hrací plochy (Base) a 0 je zachytávací objekt (Catch). Číslování dalších objektů je povoleno od čísla jedna po devět. Dalším kritériem, které je nutné splnit, je jedinečnost identifikátoru. Definice textur není povinná (samotný element textures je však povinný). Pokud se v této sekci nenalézá objektu přiřazená textura, pak se hledá textura v definici materiálů objektu a pokud ani zde nebyla textura nalezena, pak se přebírá hodnota materiálu.



Obrázek 28: Ukázka aplikace SceneGenerator

6 Testování a pokusy

Testování byla stěžejní část této práce. Jeho úkolem bylo zjistit, zda jsou finančně a technicky dostupné BCI systémy schopny umožnit jejich použití v reálných aplikacích. Pokud by takovéto systémy byly spolehlivé, a zároveň dokázaly podat dostatečný výkon, mohly by se využít v pokročilých aplikacích, jako je například ovládání invalidního vozíku. Předmětem testování byl BCI sytém od společnosti Emotiv s názvem EPOC, který je obecně známým a dostupným zařízením. Samozřejmě nelze od těchto zařízení očekávat 100% výsledky, ale jak výsledky testů a pokusů ukážou, dokážou takovéto zařízení překvapit.

Pokusy svázané s něčím, co si člověk pouze představuje, jsou velmi obtížně realizovatelné, protože nelze s jistotou říct zda je něco vyvoláno úmyslně nebo ne. Bylo tedy nutné vymyslet řadu pokusů tak, aby jejich plnění bylo co nejjednodušší a tím alespoň částečně zamezit nechtěným chybám. Pokusy bylo nutné provádět v prostředí, které bylo bez jakéhokoliv rušení, vyvolaného okolím testovaného subjektu. Jakékoliv výraznější rušení, jako je třeba nechtěný audio nebo video záznam, by mohlo testy výrazně ovlivnit. Rušení by mohlo být myslí zpracováno jako kognitivní podnět, a BCI systém toto rušení vyhodnotil jako kognitivní akci. Kognitivním podnětem, nebo akcí, se myslí stav lidského myšlení, kdy si člověk buďto úmyslně nebo neúmyslně uvědomuje či představuje nějaký objekt nebo činnost.

V rámci této kapitoly budou popsány pokusy, jejichž úkolem bylo na konkrétních příkladech ukázat schopnosti systému EPOC a společností Emotiv dodávaného SDK. Emotiv SDK umožňuje využít sadu předdefinovaných akcí, sloužících pro demonstraci SDK. Příkladem těchto akcí, které budou v rámci testů využity, jsou:

- COG_NEEUTRAL - neutrální akce
- COG_PUSH - akce oddalování
- COG_ROTATE_LEFT - akce rotace vlevo
- COG_LIFT - akce pohybu nahoru
- COG_RIGHT - akce pohybu vpravo

Pokusy byly rozděleny do několika scénářů, které se plnily postupně, a jejich výsledky se následně zapisovaly, aby byly později vyhodnoceny a zpracovány. Scénáře pak byly rozděleny podle typu do dvou skupin, které jsou popsány v následujících kapitolách.

Testům byly podrobeny dva lidské subjekty, které plnily jednotlivé scénáře pokusů. Prvním subjektem, nazvěme jej Subjekt A, byl muž ve věku 24 let ve stavu dobrého zdraví. Druhým subjektem, dále nazvaným jako Subjekt B, byl muž ve věku 52 let. Subjekt B byl taktéž v době testů v dobrém zdravotním stavu, ale na rozdíl od Subjektu A neměl žádné zkušenosti s jakýmkoli BCI systémem.

Aplikace obsahuje sekci, umožňující zobrazování seznamu detekovaných akcí a jejich trvání. To umožnilo získat některá potřebná data, sloužící k vyhodnocení testů.

6.1 Sada testů s krychlí

První skupina tří scénářů reprezentuje testy, kde bylo úkolem subjektů vyvolat pouhou myšlenkou takový podnět, který by SDK vyhodnotil jako naučenou akci. V rámci těchto testů se subjekty soustředily na naučení konkrétních, v prvním případě tří, a v druhém dvou, akcí, které byly navázány na objekt krychle. Tento objekt byl realizován 3D krychlí zobrazenou v aplikaci, a kterou se testovaný subjekt pokoušel pohybovat, jakožto reakcí na naučenou akci. První naučenou akcí v obou testech byla akce neutrální, jež plní význam jakéhosi referenčního bodu, nebo myšlenky, která pomáhá systému s rozlišením dalších akcí.

6.1.1 První scénář

Prvním ze tří scénářů, jež oba subjekty plnily, byl test rozpoznání kognitivní myšlenky systémem Emotiv EPOC. Subjekty se mimo neutrální akce naučily další akce, které měly realizovat posun testovací krychle doprava a nahoru. Prvním krokem v realizaci scénáře testu bylo učení neutrální akce, to spočívalo v soustředění mysli na plochu s 3D krychlí uprostřed. Dalším krokem bylo učení akcí COG_RIGHT a COG_LIFT. Během učení dalších akcí se subjektu zobrazovala krychle s opakujícím se pohybem v daném směru, tedy buď doleva nebo nahoru, v závislosti na učení dané akce. Tyto pohyby měly sloužit jako předloha pro představu, kterou si subjekt pokusí během testování představit. Testovaný subjekt měl za úkol vybavit si po dobu tří minut myšlenku na doleva se pohybující krychli. Tento test se opakoval celkem 5x. Z důvodu velmi častého výskytu rušení, se za správně detekovanou akci považovala ta akce, jejíž trvání bylo delší 300ms. K tomuto měření sloužila již zmíněná sekce aplikace, která kromě výpisu detekovaných akcí zobrazovala i časy jejich trvání. Výsledkem tohoto pokusu bude pravděpodobnost úspěšné detekce chtěné myšlenky, a tedy chtěné akce.

Prvním testovaným subjektem byl Subjekt A. S ohledem na to, že Subjekt A měl již se systémem EPOC a učením akcí základní zkušenosti, předpokládal se snazší průběh během učení, ale i v průběhu testování. Učení obou akcí probíhal u Subjektu A podle očekávání, a učení nebylo nutné opakovat.

Pokus	Úspěch	Chyba	Celkem	Celkem akcí	Pravděpodobnost úspěchu
1	35	16	51	78	0,69
2	34	17	51	85	0,67
3	38	14	52	83	0,73
4	40	14	54	72	0,74
5	42	13	55	65	0,76

Tabulka 1: Tabulka výsledků testů Subjektu A

Tabulka 1 vyobrazuje získaná data. První sloupec s názvem "Pokus"reprezentuje n-tý pokus, sloupec "Úspěch"pak počet chtěných akcí. "Chyba"reprezentovala počet akcí nechtěných, konkrétně druhé naučené akce pohybující se krychle vzhůru. Chyba je v tomto

případě relativní, protože může být způsobena chybnou představou člověka, nebo i chybnou detekcí SDK. Pravdou je, že oba faktory hrají stejnou roli. Sloupec "Celkem"obsahuje součty počtů chtěných a nechtěných akcí a sloupec "Celkem akcí"obsahuje počet záznamů ze seznamu detekovaných akcí bez neutrálních akcí. Poslední sloupec obsahuje vypočtenou pravděpodobnost detekované akce k celkovému počtu relevantních akcí.

Ze získaných dat lze usoudit, že testovaný Subjekt A dokázal vyvolat správnou myšlenku, ale i přesto se objevovaly akce nechtěné. Poměrně často se objevovaly i chaotické detekce, nicméně úspěšnost správné detekce chtěné myšlenky je poměrně vysoká a to v průměru 72

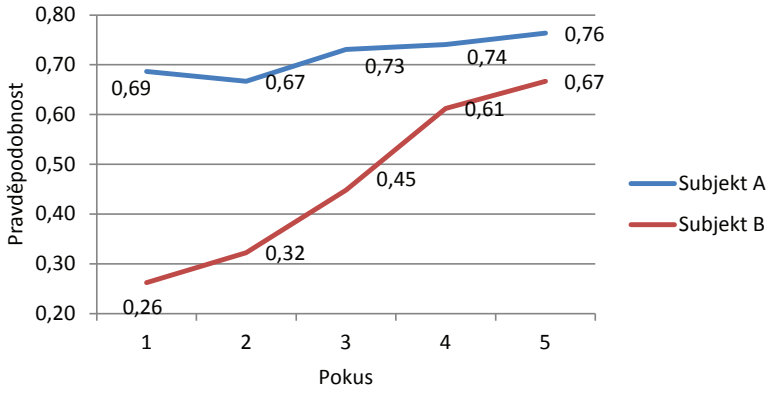
Druhý subjekt, tedy Subjekt B, měl viditelné problémy přijít na to, jak si správně vybavit konkrétní myšlenku. U tohoto subjektu proto probíhalo opakované testování, aby si subjekt B osvojil vybavování chtěných myšlenek. Každým novým učením, se subjekt B v představě zlepšoval a po deseti opakovaných učeních byl subjekt B schopen přibližně stejných výsledků jako subjekt A. V průběhu prvních pokusů, nebyl subjekt B schopen chtěně vyvolat myšlenku pohybu krychle a docházelo k chaotické detekci, která spočívala v nahodilému pohybu krychle nahoru a doprava, tudíž nešlo s jistotou říct, zda byly akce vyvolány úmyslně nebo ne. Vzhledem ke zkušenostem subjektu B, šlo spíše o nechtěné akce, nesetřízených myšlenek.

Pokus	Úspěch	Chyba	Celkem	Celkem akcí	Pravděpodobnost úspěchu
1	16	45	61	103	0,26
2	19	40	59	95	0,32
3	26	32	58	90	0,45
4	30	19	49	81	0,61
5	36	18	54	75	0,67

Tabulka 2: Tabulka výsledků testů Subjektu B

Tabulka 2 obsahující naměřená data Subjektu B, ukazují jeho snahu nalézt tu správnou míru soustředění. Subjekt B dokázal s každým dalším pokusem vyvolat správnou myšlenku s větší pravděpodobností. Zpočátku, kdy subjekt B nedokázal vyvolat tu správnou myšlenku byla detekce chaotická a úspěšnost nízká. Pátým pokusem se však Subjekt B dokázal soustředit na tu správnou myšlenku natolik, že se pravděpodobnost správné detekce blížila k výsledku Subjektu A, který měl se systémem EPOC větší zkušenosti, a tudíž věděl jak má doslova myslet.

Subjekty v průběhu testu uvedly, že nalézt tu správnou představu či myšlenku je velmi obtížné, nicméně také uvedly, že opakovaným soustředěním lze nalézt tu správnou míru soustředění a představy, která by vedla ke správné detekci. Toto lze také vyčíst z grafu 29, který zobrazuje křivku pravděpodobnosti správné detekce prvního pokusu.



Obrázek 29: Graf pravděpodobnosti úspěšné detekce

6.1.2 Druhý scénář

Druhý scénář vycházel z předešlého pokusu, ten byl rozšířen o chtěnou detekci druhé naučené akce. Cílem bylo zjistit správnost detekce dvou chtěných akcí, na rozdíl od předešlého testu, kde se subjekt snažil vyvolat pouze jednu akci. Cílem bylo zjistit zda soustředění na dvě akce ovlivní jejich správnou detekci. Učení probíhalo stejně jako v předešlém případě, tedy subjekty se naučily akce COG_NEUTRAL, COG_RIGHT a COG_LIFT. Test byl rozdělen do do dvou částí, kde v první částí se měl testovaný subjekt soustředit po dobu tří minut na akci pohybu krychle doprava. Posléze následovalo tříminutové soustředění na akci pohybu krychle nahoru. Na rozdíl od předchozího pokusu se test opakoval pouze dvakrát.

Pokus	RIGHT	Chyba	LIFT	Chyba	Úspěšnost RIGHT	Úspěšnost LIFT
1	42	14	35	20	0,75	0,64
2	41	13	38	18	0,76	0,68

Tabulka 3: Tabulka výsledků testu: Subjekt A

Pokus	RIGHT	Chyba	LIFT	Chyba	Úspěšnost RIGHT	Úspěšnost LIFT
1	38	21	33	25	0,64	0,57
2	40	19	35	22	0,68	0,61

Tabulka 4: Tabulka výsledků testu: Subjekt B

Mezi testování první akce a druhé akce byl vložen krátký interval, a to z důvodu uklidnění mysli, i přesto chvíli trvalo než si subjekt dokázal představit druhou akci. Z tabulek 3 a 4 vyplývá to, že úspěšnost detekce druhé akce je nižší než první akce. Za předpokladu, že je obtížné si představovat pouze jednu jedinou akci, je představa dvou

akcí o dost obtížnější a lze předpokládat, že s nárůstem naučených akcí roste i náročnost vybavení jejich přiřazených myšlenek a představ.

6.1.3 Třetí scénář

Další test první skupiny scénářů, byl mírně odlišný od testu předešlých. Subjekt se během testování, jež trvalo tři minuty, soustředil na vyvolání jedné jediné akce, kterou se snažil udržet co nejdéle. Testovala se schopnost člověka udržet jednu konkrétní myšlenku co nejdelší dobu, a zároveň se testovala i schopnost Emotiv SDK tuto akci detekovat co nejdéle a nepřerušovaně. Subjekt se učil myšlenku rotující krychle v jednom směru. Výsledkem testu pak měla být celková doba trvání vyvolané myšlenky. Ta se vypočte jako součet všech trvání správně zachycené akce. Tabulka 5 ukazuje výsledky testu.

Subjekt	Úspěch	Trvání
A	40	61876 ms / 62 s
B	36	52891 ms / 53 s

Tabulka 5: Tabulka výsledků testu

Test ukázal, že byly subjekty schopny udržet konkrétní myšlenku v součtu 1/3 celkového času testu. Tento interval byl samozřejmě rozložen v celé době trvání testu, a to z důvodu nemožnosti udržet totožnou myšlenku po tak dlouhou dobu, ale také z důvodu nepřetržitého působení rušení, způsobeného pokožkou hlavy, svaly nebo okolím. Nicméně se v testu ukázaly poměrně dlouhé intervaly, které trvaly i déle než 2 vteřiny, a ty lze považovat úspěch testu.

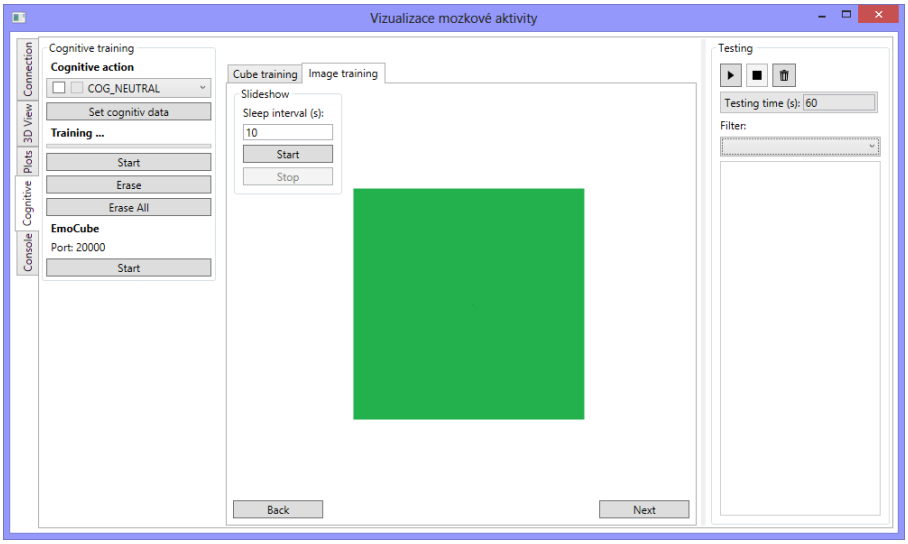
6.2 Sada testů s geometrickými tvary

Druhá skupina scénářů obsahuje sadu tří testů, jejichž úkol spočíval v rozpoznání jednoduchých geometrických tvarů. Jednoduché geometrické jsou v lidské mysli snadněji představitelné a jejich asociace s myšlenkou by měla být jednoznačnější. Úkolem následujících tří testů bylo na rozdíl od testu předešlých, kde se subjekt snažil vyvolat akci bez vizuálního podnětu, vyvolat akci vizuálně zachyceným podnětem. Jak již bylo zmíněno v testech se využívalo jednoduchých geometrických tvarů, jako jsou čtverec a kruh. Jejich jednoduchost a odlišnost by měla přispět k jejich lepšímu rozlišení v rámci asociace myšlenek. K testování slouží záložka Image training v sekci pro správu kognitivních akcí, kde lze promítat obrázky umístěné ve složce aplikace, a to buď ručně nebo automaticky v nastaveném intervalu.

6.2.1 První scénář

První scénář, jemuž byl každý testovaný subjekt podroben v rámci testů s geometrickými tvary, byl tvořen sekvencí tří vizuálních podnětů. Prvním podnětem byla čistá bílá plocha, jež měla reprezentovat neutrální akci a myšlenku. Druhým podnětem byl zelený čtverec na bílém pozadí, na který se mapovala akce SDK COG_RIGHT a třetím podnětem byl

černý kruh na bílém pozadí, na němž se mapovala akce COG_LIFT. Testovací scénář byl tvořen sekvencí těchto tří podnětů, které se v pravidelném intervalu střídaly (čtverec - bílá plocha - kruh - bílá plocha). Tento interval měl trvání 20s a celý test trval 320s. Během testu se sbíraly vzorky tvořené detekovanými akcemi, podobně jako tomu bylo v předchozích pokusech a testech.



Obrázek 30: Ukázka sekce aplikace pro učení obrazů a tvarů tvarů

1.Sekvence	**Úspěch**	**Chyba**	**2.Sekvence**	**Úspěch**	**Chyba**	**Pravděpodobnost**
čtverec	5	2		6	3	0,74
kruh	4	2		4	1	0,7
3.Sekvence			**2.Sekvence**			
čtverec	6	2		6	1	
kruh	4	2		4	2	

Tabulka 6: Tabulka výsledků testu Subjektu A

Tabulka výsledků testu 6 je rozdělena na 4 sekce, z níž každá obsahuje výsledky jednotlivých sekvencí. Z tabulky plyne fakt, že během 20s zobrazení zeleného čtverce, systém detekoval v průměru 6x příslušnou akci a 2x akci přidruženou černému kruhu. Podobný vztah se pak nalézá i u zobrazeného kruhu, jež byl v průměru detekován 4x s průměrnou chybou 2 čtverců. Celková pravděpodobnost úspěšné detekce se pak u čtverce pohybovala okolo 74% a u černého kruhu okolo 70%, což je poměrně vysoká šance.

1.Sekvence	Úspěch	Chyba	2.Sekvence	Úspěch	Chyba	Pravděpodobnost
čtverec	4	3		5	3	0,69
kruh	3	2		4	1	0,68
3.Sekvence			**2.Sekvence**			
čtverec	6	2		5	1	
kruh	5	3		3	1	

Tabulka 7: Tabulka výsledky testu Subjektu B

Tabulka 7, reprezentující výsledky testu Subjektu B se nijak výrazně neliší od výsledků Subjektu A. Ukazuje že i Subjekt B byl schopen vnímat geometrický tvar a přiřadit k němu patřičnou akci.

6.2.2 Druhý scénář

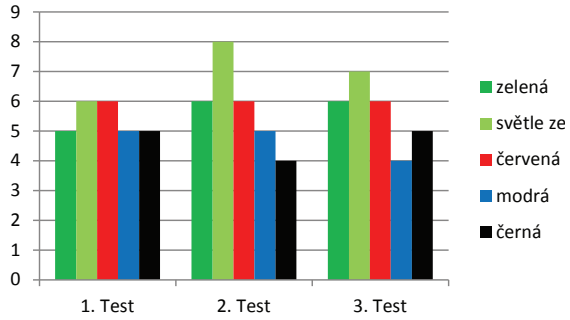
Poslední scénář měl za úkol otestovat závislost asociace myšlenky a tvaru v závislosti na barvě. Na rozdíl od předešlého testu se subjekty soustředily pouze na objekty různobarevných čtverců. Jako testované barvy byly vybrány: zelená, světle zelená, červená, modrá a černá. Obecně se ví, že lidské oko je nejcitlivější na vlnovou délku barevného spektra 555nm, kterou reprezentuje světle zelená barva. Testovaný subjekt se na každý obrazec soustředil po dobu 20 vteřin a za tuto dobu se pořídil záznam detekovaných akcí. Tento test se následně prováděl 3x.

	1. Test	2. Test	3. Test
Zelená	5	6	6
Světle zelená	6	8	7
Červená	6	6	6
Modrá	5	5	7
Černá	5	4	5

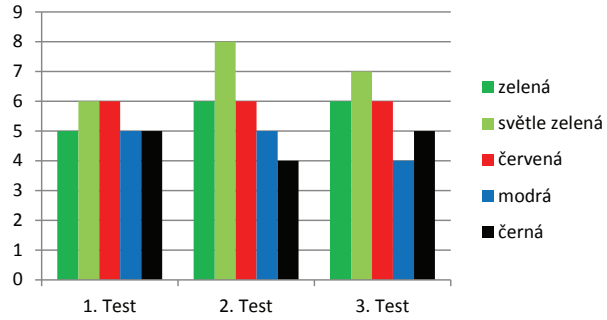
Tabulka 8: Subjekt A

	1. Test	2. Test	3. Test
Zelená	4	5	5
Světle zelená	5	6	7
Červená	6	5	6
Modrá	4	5	5
Černá	5	5	5

Tabulka 9: Subjekt B



Obrázek 31: Subjekt A



Obrázek 32: Subjekt B

Test ukázal, že změna barvy má jisté působení na schopnost mozku rozpoznat geometrický tvar. Správná barva dokáže lépe či hůře stimulovat vnímání objektu. Světle zelená barva se ukázala jako výrazný stimulant a zvýšila úspěšnost detekování příslušné akce.

6.2.3 Třetí scénář

Poslední scénář, který byl v rámci testování vytvořen, zahrnoval zakomponování snímků geometrických tvarů mezi snímky obrázků. Testované subjekty se opět snažily naučit systém asociovat svou myšlenku na viděný vizuální podnět na určité akce. Vizuální podněty byly opět čtverec a kruh. Neutrální akce byla opět bílá plocha, která se ale v rámci promítaných snímků nenalézala. Subjektům se v intervalu 20s promítalo celkem 6 snímků včetně těch testovaných. Sekvence snímků byla následující: obraz, obraz, čtverec, obraz, obraz, kruh. Celý scénář se pak opakoval 2x.

	O	O	Č	Č - chyba	O	O	K	K - chyba
1. Test	5	4	4	2	5	5	5	3
2. Test	4	3	6	2	3	4	5	3

Tabulka 10: Tabulka výsledků testu Subjektu A

	O	O	Č	Č - chyba	O	O	K	K - chyba
1. Test	6	5	5	3	5	4	4	3
2. Test	5	3	5	2	4	5	5	2

Tabulka 11: Tabulka výsledků testu Subjektu B

6.2.4 Pokusy se hrou

Poslední pokus, který byl v rámci této práce proveden, byl zaměřen na doplňkovou aplikaci práce. Tou doplňkovou aplikací byla hra, jejíž úkolem bylo prezentovat reálné využití systému BCI, konkrétně tedy systému EPOC. Jak již bylo dříve zmíněno, aplikace se ovládá dvěma způsoby, a to klávesnicí, a nebo myslí. V rámci tohoto pokusu se hra ovládala myšlenkami asociovanými k dvěma akcím SDK. V raných verzích této aplikace se uvažovalo, že se aplikace bude ovládat pouhou představou člověka na směry vlevo a vpravo. Předchozí pokusy a testy však ukázaly, že pro přesnější ovládání by bylo lepší aplikaci vylepšit o pomocné vizuální prvky, které by umožnily hráči rychleji reagovat na pozici padající kuličky. Aplikace tedy byla vybavena na obou stranách obrázky, které se zobrazují na základě aktuální horizontální pozice kuličky. Pokus se rozdělil do tří částí, lišících se v zobrazování pomocných prvků.

Nejprve se testovalo ovládání hry bez těchto prvků, brzy se ukázalo, že se jedná o neobvykle obtížný úkol, což se dalo očekávat. Padající kulička často měnila svůj směr a subjekt nestačil rychle reagovat patřičnou myšlenkou.

Další test obsahoval jako vizuální pomocné prvky zelené šipky. Zobrazování těchto prvků mírně zlepšilo ovladatelnost aplikace. Zvýšila se tak šance na pohyb zachytávacího objektu a tím i šance na zachycení kuličky.

Třetí test, reprezentoval zobrazení jednoduchých geometrických tvarů místo šipek. Aplikace se stala opět o něco ovladatelnější.

	Scóre
Bez prvků	3/10
Šipky	5/10
Geometrické tvary	6/10

Tabulka 12: Tabulka výsledků ve hraní hry

6.3 Vyhodnocení pokusů

Během pokusů se oba testované subjekty shodly, že hlavním problémem správného fungování BCI systému je najít tu správnou úroveň soustředění, sloužící k co nejlepší představě a asociaci myšlenky. Dále oba subjekty uvedly, že jistým omezením je i samotné zařízení EPOC, protože se jedná o zařízení určené pro širokou veřejnost, bylo navrženo co nejlevněji, a tudíž lze očekávat jeho jistá omezení. Dalším omezením by mohlo být vyhodnocování signálů za pomoci algoritmů v dodávaném SDK, které nemusí pracovat příliš efektivně. A posledním omezení, které mohlo ovlivnit výsledky testů, bylo samozřejmě rušení, jak vnějšími vlivy nebo svalstvem hlavy, tak i tím, že samotná myšlenka sama o sobě není jenom jedna, ale v jednom okamžiku si člověk vybavuje myšlenek několik.

Testy ale ukázaly, že si systém EPOC nevede vůbec špatně. V průměru existuje 70% šance na úspěšnou detekci chtěné myšlenky a vlastně jediným problémem je zmíněné rušení, bez kterého by byly výsledky o dost lepší a detekce by byla přesnější. Z pokusů také vyplynulo, že už vizuální podnět sám o sobě vyvolává asociativní myšlenku, která je pak detekovatelná BCI systémem. Tohoto se dá využít v reálné aplikaci BCI systému.

7 Závěr

Diplomová práce se zabývala BCI systémy, a vzhledem k tomu, že se jedná o téma z velké míry medicínské, bylo nutné nastudovat mimo technické materiály také materiály medicínské, zahrnující poznatky o získávání elektrických potenciálů mozku, jejich analýzy a následného vyhodnocení kognitivních akcích.

Cílem práce bylo také vytvořit sadu aplikací, umožňujících správu BCI systému EPOC společnosti Emotiv. Tato aplikace měla také umožňovat vizualizaci mozkové aktivity, která měla vizualizovat intenzity signálů v daných bodech, v nichž se nacházely kontakty headsetu EPOC. Aplikace dále umožňuje učení kognitivních akcí a jejich detekci, které lze pomocí UDP paketů navázat na další externí aplikace. Aplikace dále umí na modelu mozku zobrazit mozkovou aktivitu a dokáže vykreslit získané intenzity signálů v grafech.

V rámci této práce také vznikla hra prezentující reálné využití zařízení EPOC, ve které se na nakloněné rovině chytá padající kulička myšlenkami. Během testování byla aplikace rozšířena o pomocné vizuální prvky, z důvodu obtížnosti ovládání. Tyto pomocné prvky umožnily snáze ovládat hru. Jako doplňková aplikace ke hře, vznikla aplikace Generátor scény, umožňující návrh a úpravy hrací plochy.

Dalším bodem této práce bylo vytvořit reálné pokusy s headsetem EPOC. Tyto pokusy ukázaly, že zařízení EPOC je velmi zajímavým BCI zařízením, které lze využít v různých reálných aplikacích. Z pokusů ale také vyplynula složitost učení, která vyžaduje nalézt správnou míru soustředění a stavu vědomí. Je zřejmé, že i samotné zařízení EPOC má svá omezení, ale i přes ně lze říct, že je EPOC teď i v budoucnu funkčním a dostupným BCI systémem. Zajímavé by bylo porovnání s konkurenčními produkty, jako jsou MindWave a MindSet nebo s BCI systémem založeným na reálném medicínském EEG.

V průběhu implementace a testování nedošlo k žádným komplikacím a diplomová práce obsahuje vše, co po ní bylo žádáno.

Pavel Ferdian

8 Reference

[1] Seminární práce z biofyziky.
http://ftplf2.agarek.com/fyzio/prvak/biofyzika/semin/helcas_eeg.php, 2004.

[2] Brain-computer interface.
http://cs.wikipedia.org/wiki/Brain-computer_interface, 2014.

[3] Radomír Čihák. *Anatomie 3*. Grada, 2., upr. a dopl. vyd. edition, 2004.

[4] Roman JAŠEK Jaromír ŠVEJDA, Roman ŽÁK. Zpracování mozkové aktivity v bci systémech.
http://trilobit.fai.utb.cz/zpracovani-mozkove-aktivity-v-bci-systemech_e498d494-fd80-4948-a8d6-f5f3720bba2d, 2012.

[5] PřF UP Olomouc RNDr. Vlasta Lungová Ph.D., Katedra zoologie a ornitologie. Stavba a funkce lidského mozku.
http://pfyziollfup.upol.cz/castwiki/?p=3265, 2012.

A Struktura datového souboru

```
1  size
2  16;12
3  objects
4  −1;model/catch.obj
5  0;model/base.obj
6  1;model/object1.obj
7  2;model/object2.obj
8  textures
9  −1;model/texture/003.jpg
10 1;model/texture/metalx.png
11 2;model/texture/tex.jpg
12 data
13 0000000000000000
14 0000000300000100
15 0000100000200000
16 0020000000000200
17 0000020000000000
18 0000000000000000
19 0200000200000000
20 0000000000020000
21 0020000020000000
22 0000020000000200
23 0000000000000000
24 0000000000000000
```

Výpis 9: Datový soubor

B Seznam příloh

DVD Obsahuje:

- Zdrojové kódy primární aplikace
- Zdrojové kódy aplikace Hra
- Zdrojové kódy doplňkové aplikace SceneGenerator
- Digitální verze textu práce